Automation nach dem ADRRM-System

Adressloses Dezentrales Reichweitenbeschränktes Redundanzbasiertes Minimalinformationssystem zur Automation von Gebäuden und Urbanen Räumen

TECHNISCHE UNIVERSITÄT DARMSTADT

Dissertation zur Erlangung
eines Doktors der Ingenieurwissenschaften (Dr.-Ing.)
im Fachbereich Architektur
der Technischen Universität Darmstadt

vorgelegt von:
Dipl.-Ing. Martin Kim
Geboren am 16. Oktober 1969
in Suwon, Südkorea

Referent:
Professor Karl-Heinz Petzinka

Korreferenten:
Professor Manfred Hegger
Professor Doktor Ludger Hovestadt

Einreichungstermin 22. Juni 2011
Prüfungstermin 18. Oktober 2011

Erscheinungsort:
Darmstadt 2011
Hochschulkennziffer D17

Kontakt: kim@kf-a.eu

Webseite: www. kf-a.eu

Danksagung

Mein Dank gilt Helge Svenshon für seine Ratschläge zur Anfertigung dieser Arbeit und Ralph Zimmermann für die Software zur Aufzeichnung der seriellen Daten.

„Hier spukts!“

Kleines Mädchen vor dem Versuchsgebäude zu seiner Mutter.

Inhaltsverzeichnis

Hinweis: Die Fußnotennummerierung erfolgt aus programmtechnischen Gründen abschnittsweise.

1 These

1.1 These

Ein Automatisierungs- und Steuerungssystem zur Anwendung in Gebäuden und urbanen Räumen, das extrem datensparsam und nutzeranonym arbeitet und parallele sowie einfache lineare Prozesse steuern kann, selbstanpassend ausgelegt und individuell beeinflussbar ist, das den Schutz der Nutzer- und Systemdaten bereits in seiner Systemstruktur über eine vollständig dezentrale Systemorganisation ohne zentrale Zugriffs- und Angriffspunkte und adresslose Systemeinheiten gewährleistet, ist realisierbar.

1.2 Erläuterung der These

Die These beschreibt ein Automatisierungs- und Steuerungssystem, das in der Lage sein soll, auf individuelle Nutzerwünsche zu reagieren. Das System soll sich auf einfache Art und Weise an neue Steuerungsanforderungen und veränderte Umgebungsfaktoren anpassen lassen und einfach erweiterbar sein. Dieses System ist vor allem für den Einsatz in halböffentlichen und öffentlichen Räumen bestimmt, in denen die Nutzer vor ortsbezogener Datenerfassung geschützt werden sollen. Die zu steuernden Umgebungen weisen eine große Anzahl von Bauteilen auf, die parallel gesteuert werden müssen, eine lineare Steuerung auf Basis vorbestimmter Abläufe wird hierbei nur in seltenen Ausnahmefällen, wie Gefahrensituationen, erforderlich sein. Das System beschränkt sich auf die Steuerung dezentral steuerbarer Infrastruktur. Das System soll selbstanpassend ausgelegt werden, um nicht auf vorbestimmte Reaktionsmodelle angewiesen zu sein. Der Schutz der Nutzer vor ortsbezogener Datenerfassung soll bereits in der Systemstruktur verankert werden. Nachträgliche Maßnahmen wie Verschlüsselung oder Anonymisierung werden als unzureichende Schutzmechanismen angesehen und sind zum Schutz der Nutzerdaten für dieses System nicht zuverlässig genug. Das System soll daher durch einen dezentralen Aufbau, ohne zentrale Steuerung und ohne zentrale Schnittstellen, vor Zugriffen Dritter geschützt werden. Über nicht individuell unterscheidbare Einheiten, das bedeutet, Einheiten ohne individuelle Adressen, soll das System auch vor Ausspähung durch externe Abtastung gesichert werden.

1.3 Relevanz der These

In diesem Kapitel wird das Potential des in der These beschriebenen Systems unter verschiedenen Gesichtspunkten dargestellt, um die Untersuchung dieses Systems zu rechtfertigen.

1.3.1 Relevanz der These in Bezug auf die Architektur

Die Freiheit des Entwurfs kann sich nur innerhalb der Grenzen des Verantwortbaren bewegen. Sind die für die Umsetzung notwendigen Mittel nicht verantwortbar, dürfen auf ihnen basierende Planungen nicht realisiert werden. Architekten benötigen daher Mittel, deren Einsatz sie verantworten können und die sie demzufolge in ihrer Entwurfsfreiheit nicht einschränken.

Der Autor ist der Überzeugung, dass Architekten eine soziale Verantwortung gegenüber den Nutzern der von ihnen geplanten Umgebungen tragen. Diese Verantwortung hat Vorrang vor den Interessen der Auftraggeber.
Dies beinhaltet unter anderem die Verantwortung für die technische Infrastruktur der von ihnen geplanten Umgebungen. Obgleich die Entscheidungen über die Eingesetzte technische Infrastruktur oftmals von Seiten der Fachingenieure getroffen wird, liegt die bauliche Gesamtverantwortung doch bei den Architekten. Demnach tragen sie letztlich die Verantwortung, dass die dort eingesetzten Technologien mit ihrer Verantwortung gegenüber den Nutzern vereinbar sind.

In Deutschland ist der Schutz der Nutzerdaten in technischen Systemen durch das Bundesdatenschutzgesetz, kurz *BDSG*, geregelt. Die relevanten Abschnitte in Bezug auf Steuerungssysteme sind vor allem die Paragraphen: *§ 3a Datenvermeidung und Datensparsamkeit*, *§ 4 Zulässigkeit der Datenerhebung, -verarbeitung und -nutzung* und *§ 4a Einwilligung*.

> „§ 3a Datenvermeidung und Datensparsamkeit
>
> Die Erhebung, Verarbeitung und Nutzung personenbezogener Daten und die Auswahl und Gestaltung von Datenverarbeitungssystemen sind an dem Ziel auszurichten, so wenig personenbezogene Daten wie möglich zu erheben, zu verarbeiten oder zu nutzen. Insbesondere sind personenbezogene Daten zu anonymisieren oder zu pseudonymisieren, soweit dies nach dem Verwendungszweck möglich ist und keinen im Verhältnis zu dem angestrebten Schutzzweck unverhältnismäßigen Aufwand erfordert.“ [1]

> „§ 4 Zulässigkeit der Datenerhebung, -verarbeitung und -nutzung
>
> (1) Die Erhebung, Verarbeitung und Nutzung personenbezogener Daten sind nur zulässig, soweit dieses Gesetz oder eine andere Rechtsvorschrift dies erlaubt oder anordnet oder der Betroffene eingewilligt hat.
> (2) Personenbezogene Daten sind beim Betroffenen zu erheben. [...]
> (3) Werden personenbezogene Daten beim Betroffenen erhoben, so ist er, sofern er nicht bereits auf andere Weise Kenntnis erlangt hat, von der verantwortlichen Stelle über
> 1. die Identität der verantwortlichen Stelle,
> 2. die Zweckbestimmungen der Erhebung, Verarbeitung oder Nutzung [...] zu unterrichten.“ [2]

> „§ 4a Einwilligung
>
> (1) Die Einwilligung ist nur wirksam, wenn sie auf der freien Entscheidung des Betroffenen beruht. Er ist auf den vorgesehenen Zweck der Erhebung, Verarbeitung oder Nutzung [...]hinzuweisen.
> [...]
> (3) Soweit besondere Arten personenbezogener Daten (§ 3 Abs. 9) erhoben, verarbeitet oder genutzt werden, muss sich die Einwilligung darüber hinaus ausdrücklich auf diese Daten beziehen.“ [3]

Es liegt letztlich in der Verantwortung der Architekten, diesen gesetzlichen Regelungen soweit wie möglich gerecht zu werden. Konkret bedeutet dies: Systeme einzusetzen, die so datensparsam und nutzeranonym wie möglich arbeiten und grundsätzlich Systeme zu meiden, die Daten erheben, verarbeiten und nutzen, wo dies nicht unbedingt erforderlich ist. Architekten dürfen keine Systeme einsetzen, die Daten ohne explizite Einwilligung der Nutzer erheben. Liegt eine solche Einwilligung vor, müssen die Nutzer über die Erhebung und Verwendung der Daten

1 Bundesministerium der Justiz: Bundesdatenschutzgesetz. (http://www.gesetze-im-internet.de/bdsg_1990/__3a.html), 26. April 2011.

2 Bundesministerium der Justiz: Bundesdatenschutzgesetz. (http://www.gesetze-im-internet.de/bdsg_1990/__4.html), 26. April 2011.

3 Bundesministerium der Justiz: Bundesdatenschutzgesetz. (http://www.gesetze-im-internet.de/bdsg_1990/__4a.html), 26. April 2011.

unterrichtet werden. Dies ist gerade in halböffentlichen und öffentliche Räumen kaum realisierbar, da die Menschen oft auf die Nutzung dieser Räume angewiesen sind und ihnen diese Nutzung bei Verweigerung ihrer Einwilligung zur Datenerhebung und Nutzung daher nicht versagt werden darf. Die Systeme können dann allerdings nicht einfach abgeschaltet oder bestimmte Nutzer aus der Erfassung entfernt werden.

Systeme, die so ausgelegt sind, dass sie auf eine Einwilligung der Nutzer angewiesen sind, schränken entweder die Bewegungsfreiheit der Nutzer oder die Verwendbarkeit des Systems ein. Welcher der beiden Fälle eintritt wird von den vor Ort gegebenen Machstrukturen abhängig sein.

Der Philosoph Alfred Nordmann und der Soziologe Rudi Schmiede vertreten in diesem Zusammenhang die Meinung:

> „Mit Netzwerken und erst recht mit ihrer technischen Gestaltung sind immer auch gesellschaftliche Machtfragen verbunden. Hier ist ein Design der Technik notwendig, das die individuelle Kontrolle über diese erlaubt (z.B. durch An- und Abschalten der Sensoren, Regulierung der Reichweite, ggfs. Verschlüsselungstechniken etc.), damit diese Technologien überhaupt gesellschaftlich annehmbar sind.“ [4]

Hauptziel dieser Arbeit ist es, ein Steuerungssystem für Gebäude und Stadträume zu entwickeln, das dermaßen datensparsam und nutzeranonym arbeitet, dass es gesellschaftlich verantwortbar ist.

1.3.2 Technische Relevanz

Der Schwerpunkt der technischen Relevanz des in der These beschriebenen Systems liegt auf dem Schutz der Nutzerdaten.

Zentral gesteuerte Systeme ohne direkte Nutzeridentifikation können als relativ nutzerdatensicher betrachtet werden, solange ihre Daten nicht mit nutzeridentifizierenden Systeme kombiniert werden. Solche Systeme bieten allerdings kaum mehr Bedienkomfort als analoge Systeme, da keine nutzerspezifischen Systemeinstellungen hinterlegt werden können.

Zentral gesteuerte Systeme mit direkter Nutzeridentifizierung sind in Bezug auf den Nutzerdatenschutz grundsätzlich als unsicher anzusehen.
Sobald ein Nutzer identifiziert ist, hinterlässt er Spuren im System, die eine Positionsbestimmung durch die Kombination von Sensordaten und Identifizierungspunkten ermöglichen. Diese Systeme bieten allerdings einen hohen Bedienkomfort, da sich Systemeinstellungen speichern lassen, die am jeweiligen Nutzerstandort automatisch aufgerufen werden können.
Die Systemsensorik solcher Systeme kann dermaßen detailliert ausfallen, *dass Nutzer bereits anhand bestimmter Nutzungsgewohnheiten identifiziert werden können. Dies kann zu einer Ablehnung des System durch die Nutzer führen.*[5]
Diese Sensordaten werden in zentral gesteuerten Systemen an einer zentralen Stelle gespeichert. An dieser Stelle sind sie zumindest dem Betreiber des Systems und dessen nicht unbedingt *zuverlässigen Administratoren zugänglich.*[6] Der Betreiber ist damit in der *Lage Bewegungsprofile zu erstellen und Daten auszuwerten.*[7] Nutzerdaten können dort *durch externe Angriffe auch in die Hände Dritter geraten.*[8] Dieses Risiko besteht insbesondere dann, wenn es Schnittstellen nach außen, zum Beispiel zum Internet, gibt. Zudem sind *auch Netzwerke ohne direkte Schnittstellen nach außen gefährdet, wenn die Mitarbeiter zum Beispiel unvorsichtig mit Datenträgern umgehen.*[9] Die Datenerfassung in nutzeridentifizie-

4 NORDMANN, Alfred, SCHMIEDE, Rudi: Ambient Intelligence. Ethische und gesellschaftliche Herausforderungen, in: Thema Forschung - Ambient Web, 1, 2007, S. 49.

5 Vgl. MITTERBAUER, Josef: Behavior Recognition and Prediction in Building Automation Systems. TU-Wien, Fakultät für Informatik (http://publik.tuwien.ac.at/fi les/PubDat_168467.pdf), 5. März 2009, S. 92.

6 Vgl. Webseite von Cyber Ark (http://www.cyber-ark.com/news-events/pr_20080619.asp), 12. Februar 2009.

7 Vgl. MITTERBAUER, Josef: Behavior Recognition and Prediction in Building Automation Systems. TU-Wien, Fakultät für Informatik (http://publik.tuwien.ac.at/fi les/PubDat_168467.pdf), 5. März 2009, S. 92.

8 Vgl. Webseite der Kaspersky Labs GmbH (http://www.kaspersky.com/de/news?id=207566390), 13. November 2010.

9 Vgl. Kaspersky Labs GmbH (http://www.kaspersky.com/de/news?id=207566365), 27. April 2011.

renden Systemen erfordert im Sinne des BDSG die Einwilligung der Betroffenen und die Aufklärung über die Verwendung der Daten. Zentrale nutzeridentifizierende Systeme verstoßen zwar nicht gegen die im BDSG erhobene Forderung:

> „so wenig personenbezogene Daten wie möglich zu erheben, zu verarbeiten oder zu nutzen“ [10],

es ist diesen Systemen ja in der Tat nicht möglich mit weniger Daten zu arbeiten. Als datensparsam kann man diese Systeme allerdings nicht bezeichnen.

Die beschriebenen Sicherheitsprobleme zentraler Systeme könnten durch eine dezentrale Systemstruktur behoben werden.
Aufgrund der inexistenten zentralen Datenerfassung in dezentralen Systemen ist ein zentraler Zugriff auf Nutzerdaten nicht möglich. Vor einem externen Angriff sind dezentrale Systeme allerdings nur sicher, wenn das *Netzwerk eine Umprogrammierung der Knotenpunkte nicht erlaubt, die in der Folge zu einer Infizierung des Systems mit Schadsoftware führen könnte* .[11]

Eine dezentrale Systemstruktur ohne zentrale Steuerung, ohne zentrale Schnittstellen und ohne umprogrammierbare Einheiten könnte aber dennoch Angriffen durch externe Systemscans ausgesetzt sein. Es existiert Angriffssoftware, die in der Lage ist, auch *funkbasierte Netzwerke auf Schwachstellen abzutasten und die so gewonnenen Informationen nutzen kann, um das System mit schädlichen Programmteile zu infizieren.*[12] Die Eliminierung der Netzknotenadressierung verhindert diese so genannten Systemscans und wurde aus diesem Grund in die These aufgenommen.
Eine dezentrale Systemstruktur eröffnet zudem einige Vorteile, die nicht direkt mit der Nutzerdatensicherheit in Verbindung stehen.
Ein erheblicher Nachteil zentraler Systeme ist eine begrenzte Systemerweiterbarkeit. Mit wachsender Systemgröße wird der zentrale Steuerungsrechner sukzessive immer stärker belastet und erreicht irgendwann einen kritischen Punkt, ab dem er eine weitere Ausdehnung des Systems nicht mehr bewältigen kann (siehe Abbildung 1).

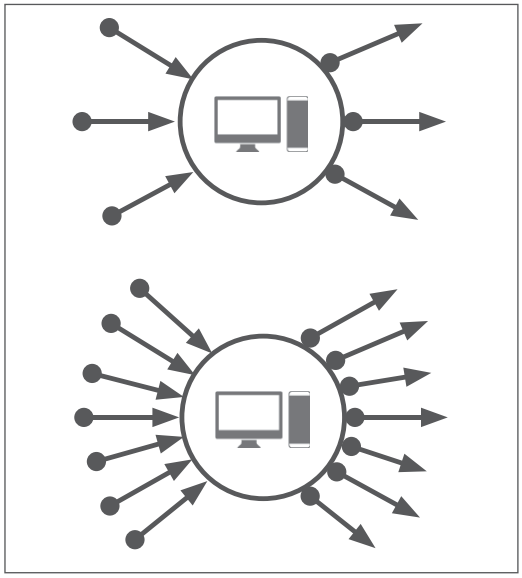


Abbildung 1:
Zentral gesteuertes System
Auswirkung einer Systemerweiterung

Dezentrale Systeme sind hingegen nahezu unbegrenzt erweiterbar. Die Rechenbelastung eines Netzwerkknotens hängt in erster Linie von dem zu verarbeitenden Kommunikationsaufkommen und der Steuerungsarbeit ab. Eine Systemerweiterung hat in dezentralen Systemen kaum Auswirkung auf diese Faktoren (siehe Abbildung 2). Die Anzahl der Nachbarn pro Knoten, die für die Kommunikationsbelastung entscheidende Größe, steigt auch im Falle einer Systemerweiterung nicht und in den Randbereichen, in denen die Erweiterung stattfindet, nur wenig an, da die Anzahl der möglichen Aktuatoren und damit die Knotendichte pro Quadratmeter relativ konstant bleibt. Dies gilt gleichermaßen für die Steuerungsbelastung, auch wenn die Anzahl der zu steuernden Aktuatoren pro Einheit im Rahmen einer Systemerweiterung nicht ansteigt.

10 Bundesministerium der Justiz: Bundesdatenschutzgesetz (http://www.gesetze-im-internet.de/bdsg_1990/__3a.html), 26. April 2011.

11 Vgl. GIANNETOS, Thanassis; DIMITRIOOU, Tassos; PRASAD, Neeli R.: Weaponizing Wireless Networks: An Attack Tool fo,r Launching Attacks against Sensor Networks. Vortrag auf der Black-Hat europe Konferenz Barcelona 2010, (http://www.ait.gr/export/sites/default/ait_web_site/faculty/tdim/various/attackTool-Black-Hat10.pdf), 11. Februar 2011.

12 Vgl. GIANNETOS, Thanassis; DIMITRIOOU, Tassos; PRASAD, Neeli R.: Weaponizing Wireless Networks: An Attack Tool for Launching Attacks against Sensor Networks. Vortrag auf der Black-Hat europe Konferenz Barcelona 2010, (http://www.ait.gr/export/sites/default/ait_web_site/faculty/tdim/various/attackTool-Black-Hat10.pdf), 11. Februar 2011.

In rein dezentral arbeitenden Netzwerken wird die ehemals zentrale Systemintelligenz bzw. die zentral organisierten Steuerungsaufgaben auf die Gesamtheit der Netzwerkknoten verteilt, dies wird als verteilte Intelligenz oder *distributed intelligence* bezeichnet.[13] Diese Verteilung der Steuerungsaufgaben bietet den Vorteil, dass die Rechenleistung und damit die Systemintelligenz analog zur Systemerweiterung mitwächst. Ein System mit verteilter Intelligenz unterliegt daher in seiner Systemgröße nicht den Beschränkungen einer begrenzten Rechenleistung, die für die Begrenzung der Systemerweiterbarkeit zentraler Systeme entscheidend ist (siehe Abbildung 2).

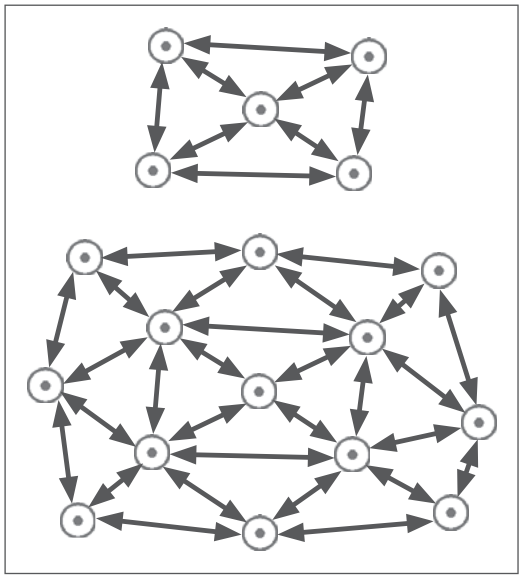


Abbildung 2:
Dezentral gesteuertes System
Auswirkung einer Systemerweiterung

Ein weiterer Vorteil dezentral gesteuerter Systeme liegt in den, im Vergleich zu zentral gesteuerten Systemen, geringeren Infrastrukturanforderungen, da bei ihnen die Steuerungsserver inklusive ihrer Räumlichkeiten entfallen.
Ein dezentrales System, das direkt in der gesteuerten Umgebung arbeitet, kann detaillierter arbeiten, da es nicht von einem zwangsläufig[14] abstrahierten Umgebungsmodell abhängig ist.

Dezentrale Systeme sind zudem äußerst ausfallsicher, da der Ausfall einzelner Einheiten nicht zwangsläufig den Ausfall des Gesamtsystems zur Folge hat. Erfolgt eine redundante Auslegung der Systemeinheiten in Anzahl und Aufgabenverteilung, kann das System auch den Ausfall mehrerer Einheiten ohne Funktionsverlust überstehen. Der Ausfall des Steuerungsservers in zentralen Systemen hingegen zieht in der Regel den Ausfall des Gesamtsystems nach sich.

1.3.3 Wirtschaftliche Relevanz

Die wirtschaftliche Relevanz des in der These beschriebenen Systems liegt, gegenüber zentral gesteuerten Systemen, hauptsächlich in der deutlich reduzierten Infrastruktur. Steuerungsserver, Serverräume und die damit verbundenen finanziellen Mittel können in dezentralen Systemen vollständig entfallen.
Die Netzwerkknoten der dezentralen Systeme, die nun die Steuerungsfunktionen übernehmen müssen, können technisch sehr einfach aufgebaut[15] und damit auch preisgünstig produziert werden.
Wird das System selbstvernetzend ausgelegt, könnten die Systemknoten ohne Hilfe von Spezialisten einfach montiert und ausgewechselt werden.
Systemerweiterungen in einem dezetralen System sind ohne Änderung der Software möglich. In zentralen Systemen müssen hierzu die Modelle der Systemsteuerung neu konfiguriert werden, was je nach Softwarestruktur, ein aufwendiger und kostenintensiver Prozess sein kann.

1.3.4 Energetische Relevanz

Die Steuerungsserver zentraler Systeme benötigen eine hohe Rechenleistung und können Abwärme produzieren, die aktiv abgeführt werden muss. Im Falle umfangreicher Systeme sind diese Server mit klimatisierten Serverschränken ausgerüstet oder in eigenen klimatisierten Räumen untergebracht. Die benötigte *Kühlenergie und Warmluftabführung beträgt in Rechenzentren bis zu 37% der benötigten Gesamtenergie.*[16]

13 pcmag.com ececlopedia (http://www.pcmag.com/encyclopedia_term/0,2542,t=distributed+intelligence&i=41566,00.asp#), 16. Februar 2009.

14 Vgl. STACHOWIAK, Herbert: Allgemeine Modelltheorie. Wien, 1973.

15 siehe: Kapitel 3.4 technischer Systemaufbau.

16 Vgl. KOCH, Peter: Weniger Kohlendioxidausstoß durch Rechenzentren mit energiesparenden Kühlsystemen. (http://www.knuerr.com/web/zip-pdf/White-Paper/de/Weniger%20Kohlendioxidausstoss-CoolTherm%20Energie-Efficiency.pdf) 17. Februar 2009.

Im Vergleich zu zentral gesteuerten Systemen, kann bei einem Einsatz dezentraler Systeme ein großer Teil der vom Steuerungssystem benötigten Energie eingespart werden. Dies betrifft den direkten Energiekonsum zentraler Steuerungsserver und vor allem auch die Energiemenge, die zur Kühlung dieser Server notwendig ist.
Die Netzwerkeinheiten dezentraler Systeme können mit Mikrokontrollern ausgerüstet werden, die einen sehr geringen Energiekonsum aufweisen und die zudem keine aktive Kühlung benötigen.

Ein dezentrales System, das direkt in der zu steuernden Umgebung arbeitet, ermöglicht ein höhere Sensordichte und kann detaillierter reagieren als ein modellbasiertes zentrales System, das nur eine begrenzte Anzahl von Sensoren berücksichtigen kann.
In einem dezentralen System kann dagegen jeder Aktuator mit einer eigenen Steuerungseinheit und eigenen Sensoren versehen werden. Dadurch kann jeder Aktuator einzeln geregelt werden und dies unabhängig von der Systemgröße. Eine detailliertere Steuerung kann zu einer signifikanten Energieeinsparung führen, da die Umgebung in kleinteiligeren Bereichen geregelt werden kann und so, systemweit betrachtet, eine energetisch optimiertere Steuerung möglich wird.

2 Stand der Technik

2.1 Systemkonzepte im Vergleich

In diesem Kapitel wird untersucht inwieweit verschiedene Systemkonzepte den Anforderungen der These entsprechen.

2.1.1 Festverschaltete dezentrale nicht programmierbare Systeme

Festverschaltete Systeme stellen die ältesten im Einsatz befindlichen Systeme dar.

In Systemen, die diesem Konzept folgen, ist die Systemlogik in einer fixierten Schaltung angelegt. In Gebäuden geschieht diese Verschaltung über eine direkte Verbindung von Schaltern und Aktuatoren oder über einfache Logikschaltkreise, wie dem Einsatz eines Relais, über das, von mehreren Schaltern aus, beispielsweise die gleichen Leuchten geschaltet werden können (siehe Abbildung 3).

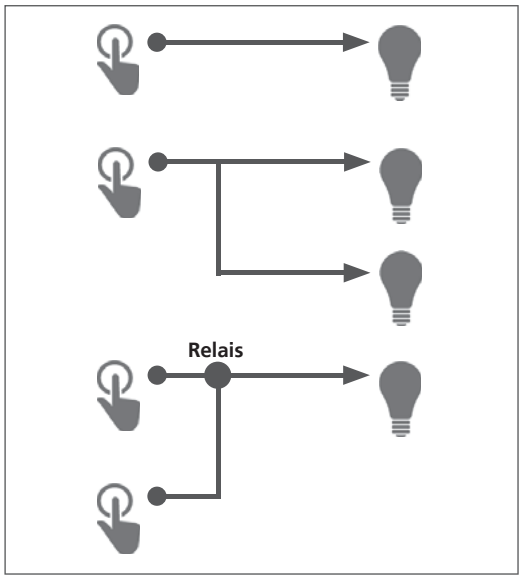


Abbildung 3:
Festverschaltete Systeme

Festverschaltete Systeme bestehen aus autonom agierenden Schaltungen. Eine Vernetzung dieser Schaltungen findet allerdings nicht statt, und ein zusammenhängendes Steuerungssystem entsteht daher nicht. Der Mangel an programmierbaren Elementen in Kombination mit der inexistenten Netzwerkstruktur verhindert den Einsatz festverschalteter Systeme zum Zwecke der Gebäudeautomation oder der Automation urbaner Räume.

Einige Forderungen der These erfüllen allerdings auch festverschaltete Systeme:

Sie besitzen keine Speicherelemente und sind somit nicht in der Lage, Nutzerdaten zu speichern, womit sie sich im Sinne der These den datensparsamen und nutzeranonymen Systemen zuordnen lassen.

Die direkte Zuordnung von Aktuatoren zu Schaltern, in festverschalteten Systemen, könnte wohl als Adressierung betrachtet werden. Diese Adressierung hat aber aufgrund der inexistenten Vernetzung und der inexistenten Datenerfassung keine Auswirkungen auf die Systemsicherheit, da weder abrufbare Daten noch Schnittstellen zum Gesamtsystem existieren.

Der dezentrale und unvernetzte Aufbau der Systeme ermöglicht zusätzlich eine unbegrenzte Systemerweiterbarkeit und die parallele Verarbeitung von Steuerungsaufgaben.

2.1.2 Zentral gesteuerte programmierbare Systeme

Bis auf eine Ausnahme, auf die im nächsten Absatz eingegangen wird, sind alle aktuellen Systeme diesem Systemkonzept zuzuordnen.

Zentral gesteuerte programmierbare Systeme bestehen aus einem zentralen Steuerungsrechner, der mit den in der Umgebung verbauten Systemaktuatoren und Systemsensoren verbunden ist. Der Steuerungsrechner befähigt das System, Abläufe automatisiert zu steuern.
Alle anfallenden Steuerungsmaßnahmen werden von diesem zentralen Rechner koordiniert (siehe Abbildungen 4, 5).

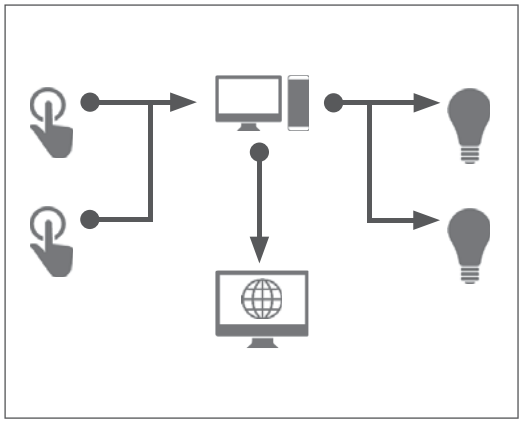


Abbildung 4:
Zentral gesteuerte Systeme

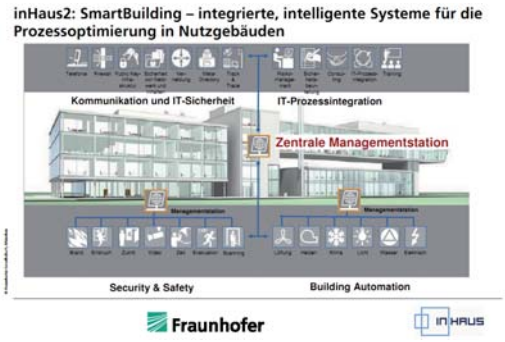


Abbildung 5 :
***inHaus* Fraunhofer Gesellschaft**

Die zentrale Organisation erfordert eine Erfassung aller steuerungsrelevanten Daten an einem Ort, um sie dort in einem Modell der zu steuernden Umgebung zu verarbeiten. Anhand der erfassten Daten werden in einem Steuerungsmodell Systemreaktionen ermittelt und in Form von Steuerbefehlen an die Systemaktuatoren übergeben.

Die Steuerungsmodelle werden von den einzelnen Steuerungssystemen gestellt und unterscheiden sich je nach Hersteller. Den Kommunikationsrahmen hingegen stellen Übertragungsprotokolle. Diese Protokolle sind in der Regel herstellerübergreifend, es existieren aber auch einige proprietäre Protokolle wie zum Beispiel *BidCos* von Contronics. Die meisten Hersteller nutzen allerdings standardisierte Protokolle, wie EIB/KNX, LON oder ZigBee. In der Folge sind die Hersteller an die Vorgaben der Protokolle gebunden. Dies betrifft vor allem den Zwang zur Adressierung der Systemelemente[1], der in allen aufgezählten Protokollen Bedingung ist.

Die zentrale Datenerfassung in zentral gesteuerten Systemen ist als elementar unsicher zu betrachten.
Es besteht grundlegend die Gefahr, dass erfasste Daten entwendet oder missbräuchlich genutzt werden. Ein Schutz der Daten auf der Ebene der Systemstruktur ist in zentral gesteuerten Systemen nicht möglich. Daten können daher in zentral gesteuerten Systemen ausschließlich durch die Überwachung der Schnittstellen und der nachträglichen Verschlüsselung der Daten oder durch die Anonymisierung der Nutzeridentäten geschützt werden. Diese Maßnahmen bieten bei gewissenhafter Anwendung zwar ein Minimum an Schutz, können aber keine Datensicherheit garantieren.
Ein Systembetreiber, der freien Zugang zu den Daten seines Systems hat, kann Nutzerdaten mißbräuchlich, das *heißt ohne Einwilligung des Nutzers und ohne seine Aufklärung über die Art der Datenauswertung zu dessen Nachteil nutzen.*[2] Systemadministratoren können ihre *Zugriffsrechte nutzen, um unberechtigt Nutzerdaten einzusehen*[3], und Daten können durch externe Angriffe auf die Steuerungszentrale entwendet oder verfälscht werden.
Auf dem Weg über eine zentrale Datenerfassungs-

1 als Systemelemente gelten Sensoren und Aktuatoren

2 Vgl. STRÖM, Pär: Die Überwachungsmafia. Wien, 2005, S. 102.

3 Vgl. Webseite von Cyber Ark (http://www.cyber-ark.com/news-events/pr_20080619.asp), 12.Februar 2009.

und Steuerungsstelle, können Schadprogramme das Netzwerk infizieren. Besonders gefährdet sind dabei Netzwerke mit externen Schnittstellen, zum Beispiel zum Internet, aber auch Netzwerke ohne Internetzugang sind gefährdet, wie *die Infizierung von Siemens Industriesteuerungsanlagen mit dem Stuxnet-Virus durch mobile Datenträger zeigt.*[4]

Zentral gesteuerte programmierbare Systeme erfüllen die Forderungen der These in den Bereichen der Fähigkeit zur Automation und Steuerung von Gebäuden und *urbanen Umgebungen*. Sie ermöglichen die Steuerung von einfachen linearen Prozessen und in gewissen Grenzen auch die Steuerung von parallelen Prozessen.[5] Zentral gesteuerte Systeme können selbstvernetzend und selbstanpassend ausgelegt werden[6] und sind individuell beeinflussbar.

2.1.3 Dezentral gesteuerte, programmierbare Systeme

In dezentral gesteuerten, programmierbaren Systemen ist die Steuerungsintelligenz auf die Knotenpunkte des Netzwerks verteilt (siehe Abbildung 6). Die Steuerungslogik dieser Systeme wird dementsprechend auch als *Verteilte Intelligenz* oder *Kollektive Intelligenz* bezeichnet.

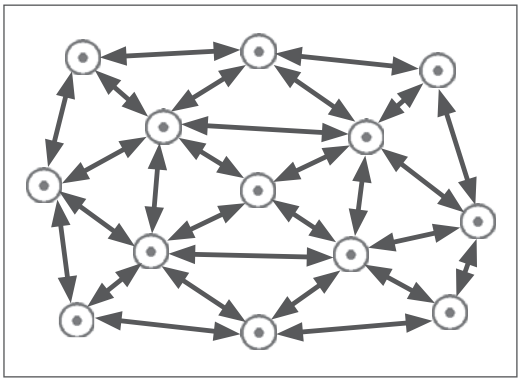


Abbildung 6:
Dezentral gesteuertes System

Es konnte nur ein einziges Steuerungssystem für den Einsatz in Gebäuden oder *urbanen Umgebung*en recherchiert werden, das diesem Steuerungskonzept folgt.[7]

Das *EnviroGird*-Steuerungsnetzwerk von *Regen Energy*[8] (siehe Abbildung 7) wurde zum Energiemanagement von halböffentlichen und öffentlichen Gebäuden und für Elektrofahrzeugladestationen entwickelt.

Dieses Netzwerk besteht aus über Funk[9] kommunizierenden Einheiten, die durch eine angeblich *auf Schwarmlogik basierenden Absprache Verbrauchspitzen in der gesteuerten Umgebung vermeiden*

4 Vgl. FALLIERE, Nicolas; O MURCHU, Liam; CHIEN, Eric: W32.Stuxnet Dossier. Version 1.3, Cupertino, USA, November 2010, S. 2.

5 Dies ist abhängig von der Rechenkapazität des zentralen Steuerungsrechners und des Abstraktionsgrades des Steuerungsmodells; generell sind zentrale Steuerungen aufgrund der begrenzten Prozessoranzahl aber nicht sehr gut zur Verarbeitung paralleler Prozesse geeignet.

6 siehe Tabelle 2.2.

7 Systeme, die über sogenannten Sensornetzwerke verfügen, sind zwar oft in der Lage, die Systemkommunikation selbstvernetzend und selbstanpassend aufzubauen, werden aber zentral gesteuert und gehören daher nicht in die Gruppe der dezentral gesteuerten Systeme.

8 Vgl. Webseite der *Regen Energy Corp.*, (http://www.regenenergy.com/default.htm), 1. Mai 2011.

9 Das *EnviroGrid*-System benutzt das ZigBee-Funkprotokoll.

und dadurch den Energiekonsum reduzieren.[10] Das System ist selbstvernetzend und im Bereich der Systemerweiterung auch selbstanpassend ausgelegt. Das *EnviroGird*-System ist allerdings ausschließlich auf Energiemanagement ausgelegt, andere Anwendungsfälle werden auf der Webseite von *Regen Energy* nicht aufgeführt.[11]

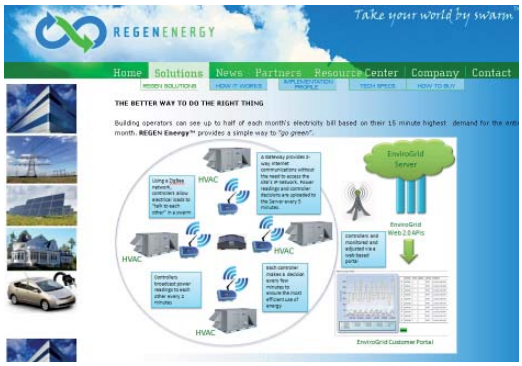


Abbildung 7:
EnviroGrid, *RegenEnergy Corp.*

Das *EnviroGird*-Steuerungsnetzwerk entspricht in den Bereichen der dezentralen Systemorganisation und der Selbstvernetzung und -anpassung den Forderungen der These. Der Schutz der Nutzerdaten ist insofern gegeben, als dass das System keine Menschen wahrnehmen kann.

In anderen Bereichen der These versagt es:

Die Automatisierungs- und Steuerungsmöglichkeiten des *EnviroGird*-Systems beschränken sich auf ein einfaches Energiemanagement und entsprechen damit nicht den Anforderungen der These nach einem individuell beeinflussbaren System.

Den generellen Datenschutzanforderungen der These wird es nicht gerecht, da es entgegen seiner an sich dezentralen Steuerungstruktur dennoch *zentral überwacht und umprogrammiert werden kann.*[12]

Die Einheiten verfügen, wie von dem verwendeten ZigBee-Protokoll vorgegeben, zudem über individuelle Adressen.[13]

10 Vgl. KULYK, Roman: Smart Grid. Taking our cue from nature, 2009. (http://www.regenenergy.com/Resources/REGEN_SmartGrid_Whitepaper.pdf), 1. Mai.2011, S. 2.

11 Vgl. Webseite der *Regen Energy Corp.*, (http://www.regenenergy.com/default.htm), 1. Mai 2011.

12 Vgl. Webseite der *Regen Energy Corp.*, (http://www.regenenergy.com/Solutions/regen_solutions.html), 1. Mai 2011.

13 KERBEL, Mark, (CEO, *REGEN Energy*): E-Mail an den Verfasser, „Yes, we use a packet protocol, and yes, each node must have its own address and we assign it.“

2.2 Eigenschaften von Beispielsystemen unterschiedlicher Systemkonzepte

Systembezeichnung	Kommunikation	Automation möglich	individuell beeinflußbar	selbstanpassend	selbstvernetzend	dezentral organisiert	ohne zentralen Zugriff	adresslose Einheiten[1]	unbegrenzt erweiterbar[2]	datensparsam	Datenschutz in Struktur
Festverschaltete Systeme	Kabel	(-)[6]	x	-	-	x	x	(-)[7]	x[8]	x	x
Zentral gesteuerte Systeme:											
Beckhoff	Bus./Funk	x	x	-	-	-	-	-	-	-	-
Digitalstrom	Stromnetz	x	x	-	x	-	-	-	-	-	-
Contronics *HomeMatic* (BidCoS)	Bus./Funk	x	x	-	-	-	-	-	-	-	-
Fraunhofer Gesellschaft *inHaus 2*	Busleitung	x	x	-	k.A.	-	-	-	-	-	-
Honeywell *Enterprise Building Integrator*	Bus./Funk	x	x	-	k.A.	-	-	-	-	-	
Siemens *APOGEE* (BACnet/ZigBee)	Funk	x	x	-	x.	-	-	-	-	-	-
Smartlighting	Infrarot Led	x	x	-	x	-	-	-	-	-	-
RWE *Smart Home*	Funk	x	x	-	x	-	-				
Dezentral gesteuerte Systeme:											
„Regen Energy“ - (ZigBee)	Funk	x[3]	-	x	x	x	-	-	x	-	-
ADRRM	Funk	x	x	x	x	x	x	x	x	x	x
Busprotokolle											
LON	Bus./Funk	x	x	(-)[4]	(-)[4]	-	-	-	(-)[4]	-	-
EIB/KNX	Busleitung	x	x	(-)[4]	(-)[4]	-	-	-	(-)[4]	-	-
ZigBee	Funk	x	x	(-)[4]	(x)[4]	(x)[4]	-	-	(-)[4]	-	-
BidCos (porprietär Contronics)	Funk	x	x	(-)[4]	(-)[4]	-	-	-	(-)[4]	-	-
ADRRM	Funk	x	x	x	x	x	x	x	[x]	x	x

[1] Systeme mit rein softwarebasierter Signalausbreitungsbegrenzung sind auf adressierbare Einheiten angewiesen; Systeme mit pysikalischer Signalausbreitungsbegrenzung nicht
[2] Systeme mit zentraler Steuerung gelten generell als nur begrenzt erweiterbar, da die Steuerungszentrale nur eine bestimmte Systemgröße verkraften kann
Systeme mit dezentraler Steuerungs sind generell unbegrenzt erweiterbar, da hier die Rechenleistung mit der Systemgröße mitwächst
[3] reines Energiemanagementsystem, kann keine weiteren Automatisierungsaufgaben übernehmen und kann nicht auf Nutzer reagieren
[4] Eigenschaft ist abhängig von übergeordnetem Steuerungssystem
[5] zentral gesteuerte Systeme gelten aufgrund der zur zentralen Steuerungs benötiigten Datenerfasung als generell nicht datensparsam
[6] theoretisch ist eine Automatisierung über festverschaltete System möglich, aber völlig unpraktikabel
[7] Systeme dieser Kategorie werden nicht als Netzwerke ausgeführt, eine Andressierung erfolgt nur lokal zwischen Schalter und Aktuator.
[8] Die theoretisch unbegrenzte Erweiterbarkeit sollte aus Sicherheitsgründen nicht voll ausgeschöpft werden, siehe: Kapitel 5.2.2.

Quellen - gesichtet 21. März 2011

Digitalstrom:	http://www.digitalstrom.org/, email an Autor
Honeywell:	http://www51.honeywell.com/hbs/losungen-sub/gebaude-sub/systeme-sub/systemintegration.html?c=31
HomeMatic:	http://www.homematic.com/, email an Autor
Fraunhofer inHaus:	http://www.i-r.de/downloads/Fraunhofer-inHaus.pdf
Smartlighting:	http://www.bu.edu/smartlighting/
Regen Energy:	http://www.regenenergy.com/, email an Autor
Siemens APOGEE:	http://www.buildingtechnologies.siemens.com/bt/us/SiteCollectionDocuments/sbt_internet_us/products-systems/building-automation/apogee/149-944.pdf
RWE Smart Home:	http://www.rwe.com/web/cms/de/455660/rwe/innovationen/energieanwendung/smarthome/
LON:	http://www.lonmark.de/lonmark-deutschland/
EIB/KNX:	http://www.knx.de/
ZigBee:	http://www.zigbee.org/

3 Verifizierung der These

3.1 Metamodell

Die Verifizierung der These erfordert zunächst ein Funktionsmodell, in dem noch, ohne Berücksichtigung der technischen Machbarkeit, geklärt wird, ob die Anforderungen der These überhaupt in einem kohärenten System fassbar sind.
Dieses übergeordnete Modell wird in dieser Arbeit als Metamodell bezeichnet.

3.1.1 Hotelmetapher

Um die Funktionsabläufe und die Organisation des Automatisierungs- und Steuerungsmodells zur Verifizierung der These anschaulich darzustellen, wird das System zunächst anhand eines metaphorischen Metamodells entwickelt.
Die Aufgaben eines Systems, die zu steuernde Umgebung und die Situation, in der sich die Nutzer befinden, lassen sich mit einem luxuriösen Hotel vergleichen. Die Nutzer übernehmen dabei die Rolle der Hotelgäste, und das Automatisierungssystem wird durch die Bediensteten der Hotels repräsentiert.

Service - Automation

Eine Besonderheit dieses Hotels besteht in der Übernahme aller umgebungsbeeinflussenden Maßnahmen durch das Personal. Den Gästen wird die gesamte Bedienung der Infrastruktur abgenommen. Die Gäste bekommen jeden Wunsch, den sie äußern, von den Bediensteten erfüllt, solange diese Wünsche innerhalb vernünftiger Grenzen liegen. Der Vorteil für die Gäste liegt bei dieser Organisationsform in einem intuitiven Umgang mit der Umgebung, ohne Kenntnisse über die technischen Möglichkeiten des Hotels besitzen zu müssen. Auf welche Weise die Wünsche der Gäste am sinnvollsten umsetzbar sind, ist Sache der Bediensteten. Diese kennen sich mit der Infrastruktur bestens aus. Aufgrund der großen Zahl der Bediensteten bildet sich ein engmaschiges Netz. Es befindet sich daher jederzeit mindesten ein Bediensteter in Reichweite des Gastes. An jedem bedienbaren Element ist ebenfalls ein Bediensteter postiert (siehe Abbildung 8).

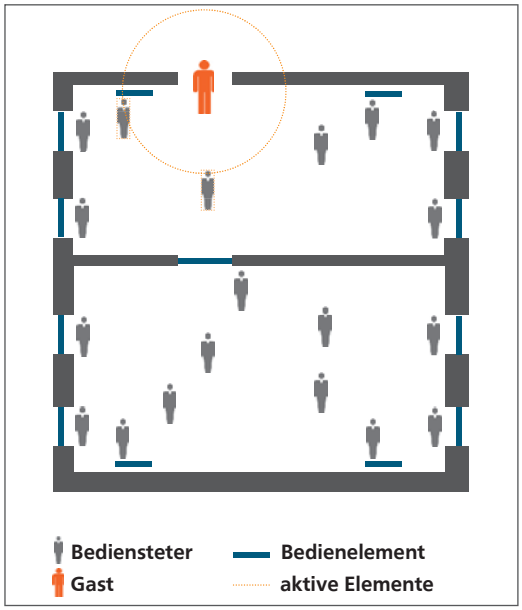


Abbildung 8:
Hotelmetapher - Gast und Bedienstete

Kommunikation

Der Gast kommuniziert ausschließlich verbal mit den Hotelangestellten, deren Namen er nicht kennt und nicht kennen muss. Die Reichweite der Stimme des Gastes bestimmt dabei seinen Einflussbereich (siehe Abbildung 9).

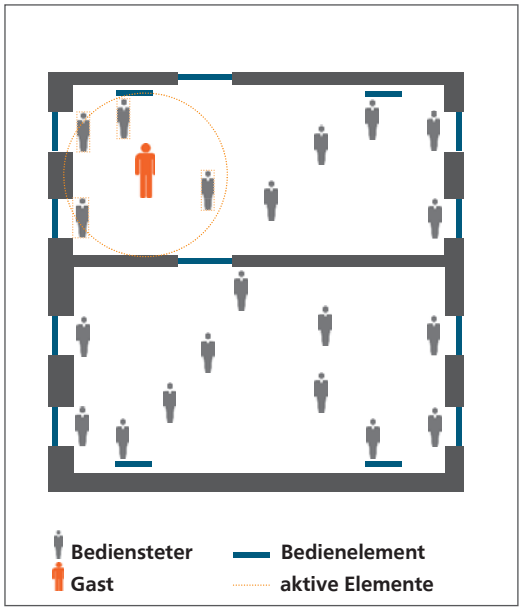


Abbildung 9:
Metamodell - Einflussbereich des Gastes

Der Gast spricht keinen der Bediensteten direkt an; wer seine Wünsche erfüllt, interessiert ihn nicht. Der Gast erwartet auch keine direkte Ansprache des Personals. Die Befriedigung seiner Anforderungen ist ihm Rückmeldung genug. Er äußert seine Wünsche allerdings so lange, bis sie in Erfüllung gehen.
Da sich auch die Mitarbeiter des Hotels aus Diskretionsgründen gegenseitig keine Rückmeldung über empfangene Nachrichten geben, übermitteln sie ihre Nachrichten mehrere Male, um eine gewisse Kommunikationssicherheit zu gewährleisten. Ist keine Reaktion der anderen Mitarbeiter zu beobachten, übermitteln sie die Nachricht noch einmal mit etwas lauterer Stimme. Diesen Vorgang wiederholen sie, bis eine Reaktion erfolgt. Die Bediensteten reagieren nur auf mehrmals mitgeteilte Nachrichten, um Missverständnisse auszuschließen, sie kommunizieren redundant.

Organisation

Sind Anforderungen eines Gastes nicht von den Bediensteten in seiner verbalen Reichweite allein erfüllbar, sprechen sich diese mit weiteren Kollegen ab. Dies geschieht über die Weitergabe des Wunsches von einem Bediensteten zum nächsten. Um nicht alle Angestellten des Hotels mit diesem einen Wunsch zu beschäftigten, wird die Weitergabe auf eine bestimmte Anzahl von Übermittlungen begrenzt (siehe Abbildung 10).

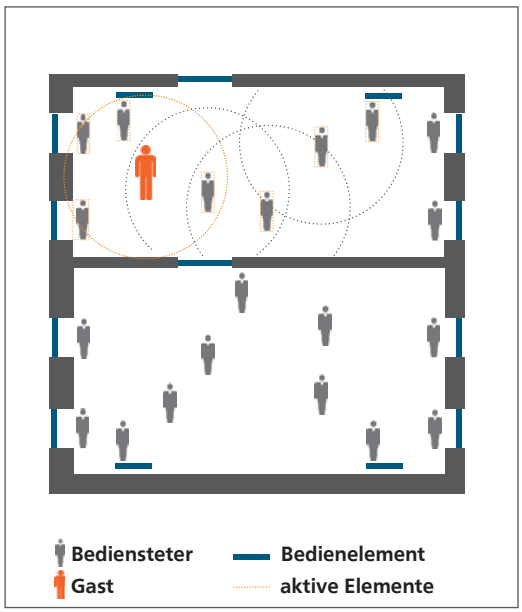


Abbildung 1o:
Hotelmetapher - Anforderungsweitergabe

Notfälle

In Notfallsituationen sollte das gesamte Hotel in Alarmbereitschaft versetzt werden. Im Gefahrenfall nutzen die Bediensteten einen mit der Weitergabe der Gästewünsche identischen Kommunikationsmechanismus, allerdings ohne Beschränkung der Weitergabeschritte (siehe Abbildung 11).

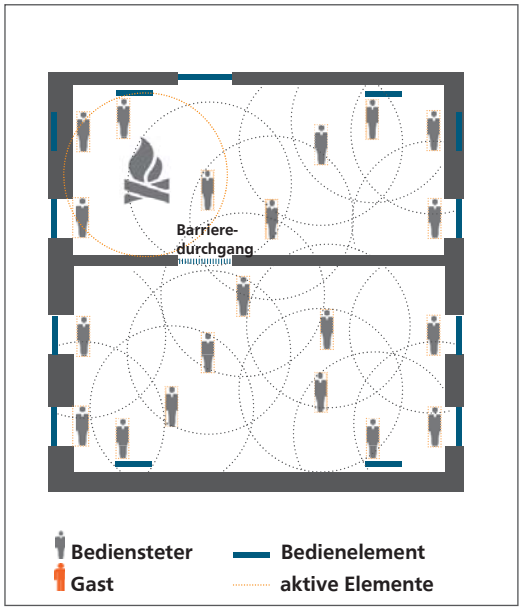


Abbildung 11:
Hotelmetapher - Notfallaktivierug

Eine permanente Alarmierung durch rundlaufende Signale wird vermieden, indem Angestellte ausschließlich Signale weiterleiten, die sie nicht bereits innerhalb eines bestimmten Zeitraumes schon einmal übermittelt haben. Die Gefahrensignale bestehen einzig aus der Nennung der Gefahrenursache. Handlungsanweisungen müssen den Angestellten nicht übermittelt werden, da sie bereits vor ihrem Einsatz in Kenntnis gesetzt wurden, wie sie in welcher Notfallsituation zu reagieren haben.
Eine Schwierigkeit der physikalischen Signalreichweitenbeschränkung ergibt sich im Falle physikalischer Barrieren. In diesem Modell erfolgt die Kommunikation über akustische Signale. Die Stimmen der Personen werden von stärkeren Barrieren, wie zum Beispiel Wänden, blockiert. Um im Gefahrenfall auch Personen hinter einem Hindernis erreichen zu können, ergeben sich zwei Möglichkeiten: Das Signal soweit zu verstärken, dass es die Barriere durchdringt oder durchlässige Elemente in das Hindernis einzufügen. Diese Problematik ist im Falle physika-

lisch reichweitenbeschränkter Systeme strukturbedingt und betrifft alle physikalischen Signale, seien es akustische, optische, olfaktorische oder elektromagnetische.

Kompromisse

Eine weitere Schwierigkeit tritt bei Anwesenheit mehrerer Gäste mit unterschiedlichen Wünschen auf. Im ungünstigsten Fall widersprechen sich die Anforderungen der Gäste, und die Angestellten sind gezwungen, einen Kompromiss finden.
Das Ziel des Kompromisses ist es, jedem Gast den maximalem Komfort an seinem Standort zu bieten. In der Folge werden mehrere Bereiche über unterschiedliche Kompromisse geregelt. Hält sich in einem Bereich eine größere Anzahl von Gästen auf, die es beispielsweise wärmer mögen, wird dieser Bereich stärker beheizt als ein angrenzender Bereich, in dem die Gäste eine kühlere Umgebung bevorzugen (siehe Abbildung 12).

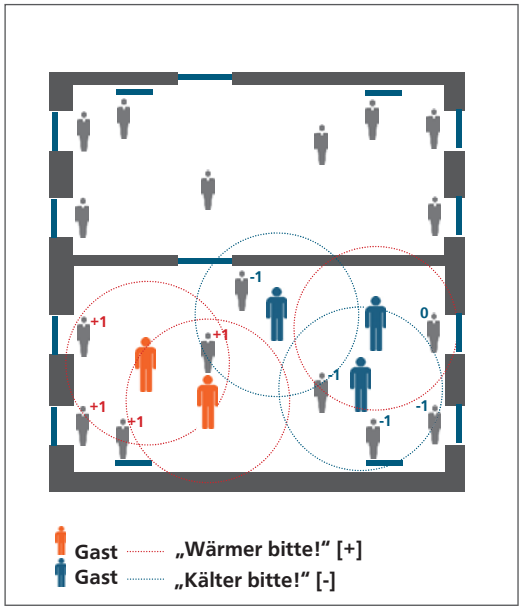


Abbildung 12:
Hotelmetapher - Mittelwertbildung

Diskretion

Die Gäste dieses Hotels stellen hohe Ansprüche an die Diskretion. Zur Gewährleistung der Diskretion ist es den Angestellten untersagt, mit Außenstehenden zu sprechen. Zudem dürfen sie die Namen der Gäste auch untereinander nicht erwähnen, und sie bekommen nur die allernötigsten Informationen über die Vorgänge innerhalb des Hotels. Die Angestellten sprechen sich auch untereinander nicht mit Namen an, um zu verhindern, dass mithörende Dritte ermitteln können, welchem Bediensteten welche Aufgabe übertragen wurde oder wo er verortet ist. Eine gezielte Ausspähung bestimmter, für sensible Bereiche zuständiger Bediensteter ist somit unmöglich.

Sicherheit

Die Sicherheit des Hotels wird von zwei Seiten gefährdet: Die Angestellten können von Dritten schädigende Anforderungen erhalten, oder sie bekommen falsche Handlungsanweisungen, nach denen sie sich in bestimmten Situationen richten könnten.
Die Gefahr durch schädigende Anforderungen ist schwer zu bekämpfen, da diese sich nicht direkt von den Anforderungen der Gäste unterscheiden lassen. Als grundlegende Abwehmaßnahme können die Bediensteten generell keine schädigenden Handlungen ausüben. Als weitere Maßnahme lässt das Sicherheitspersonal keine Außenstehenden, die nicht angemeldet sind, in das Hotel. Das Sicherheitspersonal findet seine technische Entsprechung in Zutrittskontrollsystemen. Aus den genannten Diskretionsgründen darf das Sicherheitspersonal keine Kenntnisse über die Vorgänge im Hotel besitzen, es muss als Sicherheitseinheit an den äußeren Begrenzungen des Hotels eingesetzt werden und darf nicht in Kontakt mit den übrigen Bediensteten des Hotels stehen.
Die Gefahr falscher Handlungsanweisungen, die an die Angestellten ausgegeben werden könnten, wird unterbunden, indem die Angestellten generell keine Handlungsanweisungen annehmen. Die Angestellten werden vor dem Einsatz geschult und wenden ausschließlich die vor ihrem Einsatz erteilten Handlungsanweisungen an. Falls sie später in Situationen kommen sollten, für die sie noch keine Anweisungen erhielten, versuchen sie ihre Handlungen selbständig anzupassen.

Anpassung

Es existieren zwei grundlegende Möglichkeiten der Anpassung an veränderte Umgebungsbedingungen: Das Erlernen neuer Verhaltensweisen und die evolutionäre Veränderung von Verhaltenweisen.

Lernen bedeutet die zielorientierte *Optimierung von Verhaltensweisen aufgrund von Erfahrung.*[1] Das heißt, dass die Ergebnisse der eigenen Verhaltensweisen auf ihre Effektivität bezüglich des gewünschten Ergebnisses hin überprüft werden müssen. Wurde eine Optimierung des Handlungsaufwands in Bezug zum erzielten Ergebnis erreicht, wird diese Verhaltensweise dauerhaft übernommen.
Im Falle der hochentwickelten Steuerungseinheiten des Metamodells, des Menschen, ist das Erlernen neuer Verhaltenweisen sicher die vorteilhaftere Variante. Menschen sind aufgrund ihrer sehr feinen Sensorik in der Lage, zwischen den Ergebnissen der eigenen Verhaltensweisen und den Ergebnissen zufälliger Umweltveränderungen oder den Ergebnissen der Handlungen anderer Menschen zu unterscheiden. Die damit einhergehende gute Wahrnehmung ermöglicht ihnen erst Erfahrungen zu sammeln und damit die eigenen Handlungsweisen zu beurteilen.
Systeme mit Einheiten, denen es sensorsich nicht möglich ist zwischen den Ergebnissen der eigenen und den Ergebnissen fremder Handlungen zu unterscheiden, sind nicht in der Lage Erfahrungen zu sammeln und können daher kein Lernverhalten entwickeln.
Eine Anpassung über evolutionäre Entwicklung erfodert hingegen keine sensorische Erfassung des Erfolgs des eigenen Verhaltensweisen. Aus diesem Grund können die Einheiten relativ einfach aufgebaut sein. Im Falle einer evolutionären Entwicklung zeigt sich der Erfolg der eigenen Verhaltenweisen an der eigenen Überlebensfähigkeit. Diese Überlebensfähigkeit wird über Toleranzen gegenüber den Umgebungsbedingungen geregelt. Bleiben die Umgebungsbedingungen über einen längeren Zeitraum außerhalb der Toleranzwerte, *stirbt* die Einheit und mit ihr ihre Verhaltensweisen. Eine andere Einheit mit anderen Verhaltensweisen wird ihren Platz einnehmen. Falls diese ihre Umgebungsbedingungen durch ihr veändertes Handeln innerhalb ihrer Überlebenstoleranzen halten kann, bleibt die Einheit *gesund* und kann ihre Verhaltensweisen weiter geben.
Einfachen Einheiten können sich somit einzig über evolutionäre Verhaltensänderungen anpassen.

Erweiterte Wahrnehmung

Manche Bedienstete benötigen Informationen über Umweltfaktoren, zu deren Wahrnehmung sie nicht selbst fähig sind. Dies kann seine Ursache darin haben, dass sie nicht über die notwendigen Messgerätschaften verfügen oder die maßgeblichen Umweltgegebenheiten an ihrem Standort nicht erfassbar sind. Diese Angestellten sind daher auf die Übermittlung dieser Umweltinformationen von Seiten anderer Bediensteten angewiesen. Diese stehen zum Beispiel auf dem Dach und verfügen über ein Windmessgerät. Diese zur Beobachtung postierten Angestellten geben ihre Informationen von Mitarbeiter zu Mitarbeiter in bestimmten Zeitintervallen weiter und verbreiten sie so im ganzen Hotel. Ihre Informationen erreichen auf diese Weise alle Angestellten.
Die Beurteilung der Relevanz dieser Informationen für eine Änderung bestimmter Verhaltensweisen wird von jeder einzelnen Einheit für sich getroffen.

Redundanz

Das beschriebene Hotel ist komplett dezentral organisiert und Personalausfälle sind daher nicht sofort feststellbar. Um einen reibungslosen Betrieb zu gewährleisten, muss in dieser Organisationsform mehr Personal zu Verfügung stehen, als im Idealfall notwendig wäre. Personalausfälle können somit von in Reserve stehenden Mitarbeitern ausgeglichen werden.
Bedienstete, denen es nicht gelingt, ihre Umgebung innerhalb ihrer Überlebenstoleranzen zu halten, erkranken und fallen aus. Jeden Morgen wird das Personal einem Zählapell unterzogen. Sind dabei Ausfälle in signifikantem Umfang zu vermerken, wird Ersatzpersonal entsandt. Die neuen Mitarbeiter reagieren aufgrund persönlicher Charaktereigenschaften etwas anders als die Angestellten, die sie ersetzen. Eventuell können sie ihre Aufgaben aufgrund dieser leichten Änderungen der Verhaltensweisen besser erfüllen als ihre Vorgänger und bleiben dauerhafter gesund. Der Adaptierungsprozess über den Ersatz erfolgloser Verhaltensweisen durch ihre Verrechung mit erfolgreichen Verhaltensweisen anderer Einheiten kann als evolutionäre Anpassung interpretiert werden.

1 GAGE, Nathaniel Lees; BERLINER, David C.: Pädagogische Psychologie. 5. Auflage, München 1996, S. 230.

Ereignisbasierte Handlungen

Um den Hotelbetrieb effizient zu betreiben, reagieren die Angestellten ereignisbasiert. Handlungen werden hierbei nicht zu festgelegten Zeiten aktiviert, sondern ausschließlich wenn die Umstände es erfordern. Vorbestimmte Handlungsmuster sind in ereignisbasierten Kollektiven nicht notwendig. Ereignisbasierte Systeme reagieren in Echtzeit und emergent auf veränderte Umweltfaktoren.
Aus den einzelnen Reaktionen der Mitarbeiter resultiert dann eine globale Reaktion des gesamten Hotelpersonals. *Die Entstehung globaler Zustände, die sich aus der Interaktion der Subeinheiten ergeben, wird als Emergenz bezeichnet. Hierbei besitzen die Subeinheiten kein Wissen über den Gesamtzustand.*[2]

Urbane Umgebungen

Das Hotel verfügt über eine Parkanlage[3], die mit Wasserspielen und einer Wegebeleuchtung ausgestattet ist. Um diese Anlage möglichst energieeffizient zu betreiben, soll die Wegebeleuchtung nur in den Bereichen aktiviert werden, in denen sich Personen aufhalten. Der Betreiber des Hotels legt großen Wert auf die Sicherheit der Gäste und möchte, dass zusätzlich zu der jeweiligen Leuchte am direkten Standort des Gastes auch Leuchten im näheren Umfeld geschaltet werden. Eine Aktivierung der Wasserspiele soll ebenso nur im Falle sich nähernder Gäste erfolgen. Allerdings sollen sie bereits sprudeln, wenn die Gäste sie aus der Ferne wahrnehmen können, damit für die Gäste der Eindruck entsteht, die Fontänen wären permanent in Betrieb.
Um dies zu bewerkstelligen, sind entlang der Wege Angestellte verteilt, die auf sich nähernde Gäste achten. Nähert sich ein Gast, benachrichtigt der dem Gast am nächsten stehende Bedienstete seine Kollegen in der Nachbarschaft. Die benachrichtigten Mitarbeiter geben diese Information wiederum an ihre Nachbarn weiter. Dies geschieht über drei Weiterleitungsschritte, so dass die Wegebeleuchtung in einem Radius des Abstandes dreier Parkangestellter um den aktuellen Standort des Gastes herum beleuchtet wird. Bewegt sich der Gast weiter, wird er von dem nächsten Angestellten wahrgenommen, der wiederum seine Nachbarn über das Eintreffen des Gastes informiert.
Diejenigen Angestellten, die sich nun außerhalb der Informationsreichweite befinden, schalten die Beleuchtung nach einiger Zeit selbstständig wieder ab.

Auf die Wasserspiele laufen mehrere Wege zu. Die Einheiten an diesen Wegen informieren, nach dem eben beschriebenen Prinzip, ihre Nachbarn über die sich nähernden Gäste. Erreicht diese Information den Mitarbeiter, der die Wasserspiele bedient, aktiviert dieser die Fontäne. Da die Fontäne eine gewisse Zeit braucht, bis sie ihre volle Höhe erreicht hat und beim Anfahren mehr Energie benötigt wird als im Normalbetrieb, ist es energetisch von Vorteil, sie nicht allzu oft völlig abzuschalten. Der Angestellte, der sie bedient, fährt sie daher nicht ganz runter, solange er noch Informationen über sich nähernde Gäste erhält, auch wenn sich diese noch nicht in Sichtweite befinden.

Lineare Prozesse

Diese Regelungsvorgänge zur Aktivierung der Wegebeleuchtung und der Wasserspiele können als einfache lineare Prozesse bezeichnet werden. Der Grund hierfür ist, dass die Wahrnehmung eines Mitarbeiters über mehrere Stufen nacheinander andere Mitarbeiter zu bestimmten Reaktionen bewegt.

Personalisierte Einstellungen

Da es sich um ein sehr luxuriöses Hotel handelt, ist es den Gästen nicht zuzumuten, im Falle eines Standortwechsels innerhalb des Hotels ihre individuellen Vorlieben wiederholen zu müssen. Um dem Abhilfe zu verschaffen, wird ihnen ein persönlicher Begleiter an die Seite gestellt, der ihre Vorlieben vermerkt und nach einem Ortswechsel die Mitarbeiter in der näheren Umgebung informiert. Auch dieser persönliche Bedienstete muss die Identität des Gastes nicht kennen, um seine Dienste zu verrichten. Die Diskretion über die Vorlieben einzelner Gäste bleibt auf diese Weise stets gewahrt.

2 Vgl. HASE, Christopher: Schwarmbasiertes Multipath-Routing in Sensornetzen. Norderstedt 2006, S. 13.

3 Der Park dient als Beispiel für eine urbane Umgebung, es könnte sich ebenso um eine Platzanlage oder Straßenzüge handeln, ein Park passt nur besser in die Hotelmetapher.

Bewertung des Metamodells

Eine Automatisierung von Steuerungsaufgaben eines Gebäudes oder einer *urbanen Umgebung* ist mit unbenannten Einheiten möglich. Voraussetzung hierfür ist allerdings eine physikalisch begrenzte Informationsausbreitung. Diese physikalische Begrenzung bedingt allerdings auch einige Einschränkungen. Dies betrifft zum einen eventuell auftretende kommunikative Missverständnisse. Diese Missverständnisse können durch eine hohe Informationsredundanz behoben werden. Zum anderen blockieren physikalische Hindernisse eventuell die Informationsausbreitung. Eine solche Blockade könnte durch Signalverstärkung oder speziell angelegte Durchlasspunkte überwunden werden.
Die dezentrale Organisationsstruktur bietet die Möglichkeit, diverse Aufgaben parallel zu bearbeiten, kann aber, wie im Falle der Parkanlage gezeigt, auch zur Regelung einfacher linearer Aufgaben genutzt werden.
Die Nutzer sind in der Lage die Steuerung individuell zu beeinflussen, und es besteht sogar die Möglichkeit, bestimmte Vorlieben nach eine Standortverlagerung über personalisierte Einheiten automatisiert weiter zu geben.
Die vollständige Anonymität der Nutzer und der Steuerungseinheiten ermöglicht ein sehr datensicheres System, da es keine personalisierten Daten erfassen kann.
Die dezentrale Organisation arbeitet lokal stark begrenzt, so dass nur wenige und sehr allgemein gehaltene Informationen weitergeben werden müssen, das System arbeitet daher auch extrem datensparsam.
Ein weiterer Vorteil der dezentralen und anonymen Systemorganisation ergibt sich durch die Inexistenz zentraler Zugriffs und Angriffspunkte, welche die Datensicherheit zusätzlich um Zugriffssicherheit erweitern.

Es ist festzuhalten, dass ein Steuerungssystem auf keinen Fall mit einem Zutrittskontrollsystem kombiniert werden darf. Diese strikte Trennung der Systeme ist notwendig, um die Nutzeridentität zu schützen, denn Zutrittskontrollsysteme müssen zur Erfüllung ihrer Aufgaben die Nutzeridentität feststellen. In Kombination mit der Sensorik der Steuerungssysteme wäre eine individuelle Ortsbestimmung und Wegverfolgung der Nutzer möglich, und dies gilt es zu vermeiden. Des Weiteren sollten Steuerungssysteme aus Sicherheitsgründen keine Kontrolle über die Zugänge erhalten.[4]

Das metaphorische Hotelmodell zeigt, dass es grundsätzlich möglich ist, eine Umgebung den Anforderungen entsprechend kohärent zu organisieren. Da es sich um ein rein theoretisches Modell handelt, kann es nur zeigen, dass ein der These folgendes System grundsätzlich vorstellbar ist. Es beweist nicht dessen reale Funktionsfähigkeit. Aus der Hotelmetapher lassen sich jedoch die grundsätzlichen Systemprinzipien ableiten, auf denen ein dezentrales System mit anonymen Einheiten basieren kann.
Der praktische Nachweis der Leistungsfähigkeit dezentraler Systeme auf Basis anonymer Einheiten wird im Kapitel der natürlichen Metamodelle erbracht.

4 siehe Kapitel 5.1.2 Emergente Verhaltensweisen

3.2 Vorbildsysteme

Das Hotelmetapher-Metamodell repräsentiert zwar bereits ein kohärentes Organisationsmodell. Organisationsstrategien und Kommunikationsprinzipien lassen sich aus diesem Modell allerdings nicht in ausreichendem Maße ableiten.
Die natürlichen Systeme staatenbildender Insekten und künstliche Agentensysteme könnten die benötigten Strategien und Prinzipien aufweisen.

3.2.1 Natürliche Systeme - staatenbildender Insekten

> Staatenbildende Insekten, soziale Insekten, solche Insekten, die sich zum Zweck der Brutfürsorge zusammentun, deren Nachkommen Verbände (Staaten) bilden und weiterhin für eine Nachkommenaufzucht zusammenbleiben.[1]

Staatenbildende Insekten dienen der Überprüfung der These in zweierlei Hinsicht: Sie beweisen die Leistungsfähigkeit dezentraler anonymer Systeme, und weisen grundlegende Organisations- und Strukturprinzipien auf, die zur Entwicklung eines künstlichen Steuerungskollektivs notwendig sind.

3.2.2 Leistungsfähigkeit staatenbildender Insekten

Staatenbildende Insekten bieten sich als reale Metamodelle für dezentrale anonyme Steuerungs- und Automatisierungssysteme an, da sich ihre Staatenmitglieder nicht individuell unterscheiden können und ein Insektenstaat dezentral organisiert ist.
Bienen, Ameisen und Termiten beweisen eindrucksvoll, zu welchen Leistungen Systeme, die allein auf dezentralen Organisationsprinzipien aufbauen, fähig sind. Alle drei genannten Spezies sind in der Lage, komplexe Bauten mit großer *Funktionsvielfalt, zum Beispiel einer Klimatisierung*[2] (siehe Abbildung 13), zu errichten.
Diese Staatenkollektive verfügen über dezentral organisierte Aufgabenteilung und Aufgabenbewältigung. Dies betrifft unter anderem die Futtersuche *in bis zu 500m langen Beutezügen*[3], *Brutpflege*[4], *Abwehr von Eindringlingen*[5], *Errichtung, Instandhaltung und Klimatisierung der Bauten.*[6]
Sie bewältigen damit Aufgaben von einer Komplexität, die weit über die Anforderungen einer Gebäudesteuerung hinausgehen. Dezentrale anonyme Systeme sind somit grundsätzlich leistungsfähig genug, um den Anforderungen der These gerecht zu werden. Wenn es gelingt, die Organisationsstrategien und Kommunikationsprinzipien, nach denen diese natürlichen Kollektive arbeiten, auf ein künstliches System zu übertragen, könnten diese künstlichen Kollektive die Bedingungen der These erfüllen.

Abbildung 13
Gipsausguss der Hauptventilationsröhren eines Termitenbaus der Gattung Macrotermis michaelseni

1 SAUERMOST, Rolf; FREUDIG, Doris et al. [Redaktion]: Lexikon der Biologie. Band 12-Resolvase bis Simvastatin, Heidelberg 2002.

2 Vgl. LOHMANN, Dieter: Klimaanlagen für Termiten-Türme. in: scinexx, Düsseldorf 2006, (http://www.scinexx.de/dossier-detail-292-10.html), 22. Februar 2011, S. 1.

3 Vgl. KIRCHNER, Walter: Die Ameisen: Biologie und Verhalten. München 2001, S. 39.

4 Vgl. KIRCHNER, Walter: Die Ameisen: Biologie und Verhalten. München 2001, S. 91.

5 Vgl. SUGAHARA, Michio; SAKAMOTO, Fumio: Heat and carbon dioxide generated by honeybees jointly act to kill hornets, in: Naturwissenschaften, 96, Heidelberg 2009, S. 133–1136.

6 Vgl. LOHMANN, Dieter: Klimaanlagen für Termiten-Türme. in: scinexx, Düsseldorf 2006, (http://www.scinexx.de/dossier-detail-292-10.html), 22. Februar 2011, S. 1.

3.2.3 Organisation staatenbildender Insekten

Dezentrale Organisationsstruktur

Die Aufgaben, die ein Ameisenstaat zu bewältigen hat, sind von einem breiten Spektrum von Umweltreizen abhängig. Einem einzelnen Individuum wäre die Wahrnehmung und Auswertung der Gesamtheit der bedeutsamen Reize nicht möglich. Nur das gesamte Kollektiv mit all den Wechselbeziehungen seiner Individuen vermag eine koodinierte Zusammenarbeit zu leisten.[7]

Emergenz

Staatenbildende Insekten erbringen ihre Leistungen ohne hierarchische, *top-down* genannte, Strukturen. Die sogenannten Königinnen *besitzen keinen relevanten organisatorischen Einfluss, ihre Bezeichnung ist in dieser Hinsicht irreführend.*[8]

Die Organisation staatenbildender Insekten erfolgt strukturell von unten nach oben, *bottom-up* (siehe auch 3.3.1 Systemtheoretische Grundlagen). Die von außen betrachtet globalen Verhaltensweisen ergeben sich im Falle der Insektenstaaten aus der Gesamtheit einfacher, regelbasierter und individueller Verhaltensweisen einzelner Staatenmitglieder. Dieses Phänomen wird als *Emergenz* bezeichnet (siehe auch 3.3.1 Systemtheoretische Grundlagen).

Beispiele für *emergentes* Verhalten finden sich beispielsweise in den Morphologien der Vogel- und Fischschwärme. Aus diesem Grund werden diese emergenten Phänomene auch als Ergebnis von *Schwarmintelligenz* bezeichnet. Staatenbildende Insekten arbeiten allerdings wesentlich kooperativer zusammen als Schwärme. Der Begriff der *Schwarmintelligenz* wird daher in dieser Arbeit nicht weiter genutzt. Die treffendere Bezeichnung für die Basis emergenten Verhaltens ist *Kollektive Intelligenz*.

Kollektive Intelligenz

> „Collective Intelligence is a shared or group intelligence that emerges from the collaboration and competition of many individuals.“ [9]

Insektenstaaten basieren auf *Kollektiver Intelligenz*. Eine Grundvoraussetzung, die dezentrale Systeme benötigen, ist eine Verteilung der Aufgaben auf ein Kollektiv. Anders ausgedrückt: *Zentrale Intelligenz* wird in *Kollektive Intelligenz* transformiert.

Dezentrale Aufgabenverteilung erfordert eine Zerlegung der globalen Aufgaben in regelbasierte Verhaltensweisen. Diese Regeln müssen in ihrer kollektiven und kombinierten Anwendung zu sinnvollen globalen Prozessen führen. Die hieraus entstehenden Prozesse werden ab einer bestimmten Systemgröße nicht mehr aus den Regeln ersichtlich sein und damit *emergent* erscheinen.

Die Leistungen staatenbildender Insekten zeigen das enorme Potential regelbasierter, dezentral organisierter Kollektive. Eine der Hauptaufgaben bei der Entwicklung technischer, auf *Kollektiver Intelligenz* basierender Systeme wird in der Entwicklung sinnvoller Regeln und deren Feinabstimmung bestehen. Die Qualität des Verhaltens Kollektiver Systeme hängt in erster Linie von der Qualität ihrer Regeln ab. Die Anpassung des Systems an veränderte Umgebungsverhältnisse kann in diesen Systemen einzig über eine Veränderung der Regeln erfolgen.

Die Genese *emergenter* Verhaltenweisen staatenbildender Insekten zeigt, dass von außen kompliziert erscheinende und sinnvoll ineinander greifende Vorgänge, ohne zentrale und hierarchische Steuerungsstrukturen möglich sind. In Bezug auf technische Systeme bedeutet dies, dass eine Automatisierung und Steuerung umfangreicher Umgebungen keine hierarchische und zentrale Organisationsstruktur erfordert, sondern ebenso durch ein dezentrales System erfolgen kann.

Evolutionäre Anpassung

Die relativ einfachen Gehirne der Mitglieder staatenbildender Insekten, Ameisen besitzen etwa beispielsweise nur 250.000 Neuronen,[10] ermöglichen ihnen kaum relevantes Lernverhalten. Eine Selbstanpassung der Verhaltensregeln staatenbildender Insekten kann daher nur evolutionär erfolgen (siehe Kapitel 3.1.1 Hotelmetapher und 3.4.2 Systemstruktur). Staatenbildende Insekten haben diese Verhaltens-

7 Vgl. KIRCHNER,Walter: Die Ameisen: Biologie und Verhalten. München 2007, S. 78

8 Vgl. KIRCHNER,Walter: Die Ameisen: Biologie und Verhalten. München 2007, S. 65.

9 MIT Center for Collective Intelligence: Handbook of Collective Intelligence. (http://scripts.mit.edu/~cci/HCI/index.php?title=What_is_Collective_Intelligence%3F), 11. Mai 2011.

10 Vgl. MURPHY, Nancey C. et al.: Did my neurons make me do it?. Oxford 2007, S. 99.

weisen evolutionär perfektioniert und sind aufgrund sehr gut abgestimmter Verhaltensregeln enorm leistungsfähig. Systeme, die sich einzig evolutionär entwickeln können, sind auf schnelle Generationswechsel angewiesen, um auch ohne Lernvermögen schnell genug auf Umgebungsveränderungen reagieren zu können. Die Generationswechsel natürlicher Systeme sind durch Paarungszeiten und die Lebensspanne der Tiere getaktet.
Ein technisches System mit einfachen, nicht zu Lernverhalten fähigen Einheiten, kann dementsprechend ebenfalls nur über evolutionäre Prinzipien selbst anpassend ausgelegt werden. Um sich schnell an Umgebungsveränderungen anzupassen, müssen die Generationswechsel auch in künstlichen Systemen in kurzen Abständen erfolgen. Hierzu ist auch bei künstlichen Systemen eine Generationentaktung notwendig, die eine evolutionäre Entwicklung erzwingt.

Zeitlich begrenzte Paarungszeit

Wie bei fast allen Tiere erfolgt die Paarung bzw. die evolutionäre Fortentwicklung staatenbildender Insekten in einem zeitlich eng begrenzten Bereich. Da Tiere in der Paarungszeit besonders gefährdet sind, wird diese möglichst kurz gehalten. Im Falle der Ameisen beträgt die Zeitspanne des Hochzeitsfluges nur einen Tag.
Auch künstliche Systeme sind in der Zeit ihrer evolutionären Entwicklung besonders durch Angriffe von außen gefährdet. Der Austausch von Erbgut ähnlichen Informationen kann von Angreifern genutzt werden, um schädliche Werte in das System einzuspeisen. Es ist daher sinnvoll, die evolutionäre Entwicklung auch im Falle künstlicher Systeme zeitlich eng zu begrenzen. Des Weiteren sollte der evolutionäre Prozess zur Abwehr von Angriffen zu einem zufälligen Zeitpunkt stattfinden.

Abstimmung

Eine globale Entscheidungsfindung ist in dezentralen Systemen schwierig, da eine dezentrale Abstimmung erfolgen muss. Ohne eine zentrale bewertende Stelle erscheint dies zunächst unlösbar.
Staatenbildende Insekten haben hierfür eine einfache Lösung gefunden.

> „Ist eine Ameise fündig geworden, kehrt sie zum Nest zurück, „rempelt“ eine Nestgefährtin an und fordert sie so auf, eine Haltung einzunehmen, die zum Tragen günstig ist; dann bringt sie die eingerollte Nestgefährtin zum neuen Nestplatz. Ist die getragene Ameise von der Qualität des neuen Platzes überzeugt, wird sie nach ihrer Rückkehr ins Nest ihrerseits eine andere Ameise anrempeln und zum neuen Nestplatz tragen. Durch ständige Wiederholung greift dieses Verhalten auf immer mehr Individuen über („Schneeballsystem“) und führt schließlich zum Umzug des Volkes.“ [11]

> „Potentielle Sammelbienen, die noch keine Kenntnis von Futterquellen haben, suchen gezielt diesen Bereich auf und verfolgen dort wahllos Tänze (=Empfang von Informationen), indem sie als „Nachtänzerinnen“ einer heimkommenden Sammlerin beim Tanz nachlaufen und so die Informationen aufnehmen. [...] [Auf diese Weise] entscheidet sich eine Kolonie im Experiment innerhalb einer halben Stunde nach einem Wechsel der Zuckerkonzentrationen für die neue effizientere Futterquelle [...].“ [12]

Bienen und Ameisen sind demnach in der Lage, globale Entscheidungen über eine dezentral ablaufende Abstimmung zu treffen. Dezentrale künstliche Systeme werden ähnliche Verfahren benötigen, um Entscheidungen im Falle widersprüchlicher Anforderungen treffen zu können.

Erweiterbarkeit

Die Potenz von Insektenstaaten steht in direktem Zusammenhang zu der Größe ihres Kollektivs. Ihre Stabilität und Überlebensfähigkeit ist stark von der Anzahl der Staatenmitglieder und der damit einhergehenden Redundanz abhängig. Zu ihrem großen Vorteil sind Insektenstaaten aufgrund ihres dezentralen Aufbaus in der Lage ihr Kollektiv nahezu unbegrenzt zu erweitern. Daraus resultieren teilweise enorme umfangreiche Kolonien von *Millionen zusammenhängenden Nestern über eine Länge von bis zu 6000 Kilometern.*[13]
Große Kolonien sind dementsprechend sowohl in Bezug auf die Staatenmitglieder als auch in Bezug auf den Lebensraum hoch redundant und extrem stabil. Künstliche dezentrale Systeme lassen sich theoretisch gleichfalls unbegrenzt erweitern und wer-

11 KIRCHNER, Walter: Die Ameisen: Biologie und Verhalten. München 2001, S. 74-75.

12 SCHMICKL, Thomas: Sammeln, Verteilen und Bewerten von Informationen. Verteilte Intelligenz in einem Bienenvolk, in: FACTS Die Informationsgesellschaft, 1, Wien 2003, S. 5.

13 DOWRSCHAK, Manfred; KELLER Keller: Supermacht im Untegrund, in: Der Spiegel, 17, 2002, (http://wissen.spiegel.de/wissen/image/show.html?did=22151068&aref=image029/E0217/SCSP200201702040205.pdf&thumb=false), 20. Februar 2011.

den mit zunehmender Größe ebenfalls stabiler. Dies ist jedoch nicht unbedingt nur von Vorteil, da große Systeme nahezu physikalisch unzerstörbar werden und damit praktisch nicht mehr abschaltbar sind (siehe Kapitel 5.2 Gefahren).

Redundante Mitgliederzahl

Ein Ameisenstaat umfasst mehr Mitglieder als im optimalen Fall benötigt werden. *Es existieren Individuen, die keiner spezialisierten Arbeit nachgehen und die meiste Zeit untätig verbringen. Diese unbeschäftigten und unspezialisierten Ameisen greifen immer dann in das Geschehen ein, wenn Gefahrensituationen auftreten oder andere Ameisen Unterstützung brauchen.*[14]

Auf künstliche Systeme bezogen bedeutet dies, dass ein gewisser Prozentsatz redundanter physikalischer Einheiten nötig ist. Nur so können ausgefallene Einheiten schnell ersetzt werden und in Krisensituationen mehr Systemleistung aufgebaut werden.

Gefahrenabwehr

Staatenbildende Insekten sind größtenteils physischen Angriffen ausgesetzt, die sie ebenso physisch bekämpfen. Ein physischer Angriff auf ein künstliches Steuerungssystem ist nicht zu erwarten. Es existieren allerdings Schädlinge, die von den Ameisen geduldet werden. Die Angreifer nutzen hierbei chemische Botenstoffe, sogenannte *Pheromone*[15], die die Nestzugehörigkeit anzeigen, um sich unbemerkt Zugang zu dem Ameisenstaat zu verschaffen. Gegen diese Angreifer ist ein Ameisenstaat machtlos. Die Gefahr durch Angreifer, die sich einen Zugang zum System durch eine gefälschte Kollektivmitgliedschaft verschaffen, existiert auch in künstlichen Systemen. Es ist daher sinnvoll, alle Mitgliedern eines Kollektivs mit einer verborgenen gemeinsamen Kennung zu versehen.

3.2.3 Kommunikationsprinzipien staatenbildender Insekten

Die Kommunikation der staatenbildenden Insekten erfolgt ebenfalls über physikalisch reichweitenbegrenzte Mittel. Die Reichweitenbeschränkung ihrer Kommunikationsmittel erlaubt staatenbildenden Insekten eine differenzierte Aufgabenverteilung ohne *persönliche* Ansprache einzelner Staatenmitglieder. Auf künstliche Systeme übertragen bedeutet dies eine Kommunikation ohne Adressierung der Systemeinheiten.

Kommunikation über Minimalinformationen

Ameisen kommen mit einem minimalen Satz von Kommunikationssignalen aus. Die Arbeiterinnen der Solenopsis invicta kommunizieren beispielsweise nur über zehn Worte, neun davon bestehen aus Pheromonen und eines wird taktil übermittelt.[16]

Ein Kommuniaktionssystem zur Organisation dezentraler Netzwerke kann demnach auf relativ wenigen Signalen basieren.

Beeinflussung statt Anweisungen

Pheromone beinhalten nur ein minimales Informationsvolumen, erweisen sich aber organisatorisch als sehr wirkungsvoll. Die Wirkung der Pheromone besteht oft nur in der Beeinflussung des Verhaltens der empfangenden Ameise. Die Pheromone bewirken somit nur die Verschiebung der Wahrscheinlichkeit eines bestimmten Verhaltens, sie lösen nicht zwangsweise direkt ein Verhalten aus. Diese Wirkungsweise hat den Vorteil, dass auch weitere Einflussfaktoren nur zur Verhaltensausrichtung berücksichtigt werden können. Dies ermöglicht Ameisen eine flexiblere Anpassung ihres Verhaltens an die Umwelt.[17]

Die Organisation dezentral organisierter Kollektive kann also über beeinflussende Signale statt Anweisender erfolgen.

Lokale und temporäre Signalbeschränkung

In der Hotelmetapher wurden akustische Kommunikationsmittel genutzt, im Falle der staatenbildenden Insekten handelt es sich hauptsächlich um *Pheromone*. *Pheromone* sind ebenso wie akustische Signale räumlich und zeitlich in ihrer Wirkung begrenzt. Der Signalwirkungsradius ist auf einen *Aktiveraum* bzw.

14 Vgl. KIRCHNER, Walter: Die Ameisen-Biologie und Verhalten. München, 2001, S. 67, 69.

15 FRINGS, Stephan,MULLER, Werner A.: Tier- und Humanphysiologie: Eine Einfuhrung. Heidelberg 2009, S. 582.
„Pheromone sind Signalsubstanzen, die von einem Individuum nach außen abgegeben werden und bei anderen Individuen der gleichen Art spezifische, vorprogrammierte Reaktionen auslösen."

16 Vgl. HÖLLDOBLER, Bert; WILSON, Edward: The Ants. Harvard 1990, S.288

17 Vgl. HÖLLDOBLER, Bert; WILSON, Edward: The Ants. Harvard 1990, S.253

den *active-space* beschränkt. Der *Aktivraum* gliedert sich in einen *anziehenden Randbereich* und einen *aktivierenden Bereich* um das Zentrum der Pheromonausschüttung herum. [18]
Es ist demnach entscheidend, ein technisches Kommunikationsmittel zu finden, mit dem sich ebenfalls eine physikalische Reichweitenbeschränkung umsetzen lässt.
Die Kommunikationsbeschränkung durch Reichweitenbegrenzung und temporär begrenzter Wahrnehmbarkeit ist unpräzise. *Pheromone* bestehen aus flüchtigen chemischen Substanzen, deren Ausbreitung und Flüchtigkeit nicht genau vorhersehbar ist. Diesen Ungenauigkeiten in der Kommunikation wirken Ameisen durch mehrere Maßnahmen entgegen:

Differenzierte Signalkonzentration

Zweitens ist *die Konzentration der Pheromone entscheidend für ihren Einfluss auf das Verhalten der Ameisen.*[19] *Einer frischen, hochkonzentrierten Spur wird gefolgt, einer älteren Spur mit schwacher Pheromonkonzentration nicht mehr.*[20]
Ein künstliches Kollektiv muss seine Entscheidungen somit anhand von Schwellenwerten treffen.

Redundante Signale

Drittens werden *Pheromonspuren mehrfach angelegt*[21] und erreichen höhere Konzentrationen als zur Wahrnehmung nötig wären. Auch bei der Signalübertragung von Biene zu Biene kommt es zu Kommunikationsproblemen, die über eine mehrfache Wiederholung der Signale ausgeglichen werden.

> „Wer das Gewimmel in einem Bienenstock kennt, kann sich leicht vorstellen, dass diese Kommunikation einem starken „Hintergrundrauschen“ ausgesetzt ist, und die Informationsübertragung sicherlich mit einem gewissen Maß an „Sendefehlern“, „Übertragungsfehlern“ und „Empfangsfehlern“ belastet ist. Entsprechend der Allgemeinen Informationstheorie von Shannon (vgl. Shannon 1948, S. 379-423) ist es daher nicht verwunderlich, dass die Informationsübertragung redundant erfolgt, dass bedeutet, dass die Nachtänzerinnen mehrere Tanzrunden verfolgen: Die Empfänger empfangen also mehrere verrauschte Kopien der Nachricht.“ [22]

In künstlichen Systemen muss also ebenfalls mit redundanten Signalen gearbeitet werden.

Differenzierte Signalauswertung

Viertens werden verschiedene Signale miteinander verglichen und nur das relevantere Signal, zum Beispiel ein Gefahrensignal, wird befolgt.
Technische Systeme müssen demnach ebenfalls in der Lage sein, mehrere Signale gleichzeitig wahrzunehmen, zu vergleichen und daraufhin eine Entscheidung zu treffen. Dies spricht in künstlichen Systemen für den Einsatz neuronenähnlicher Elemente.

Globale Signalverbreitung

Aufgrund der lokal begrenzten *Pheromonausbreitung* ist es Ameisen nicht möglich, direkt globale Signale zu übermitteln. *Die globale Signalverbreitung erfolgt indirekt von einem Kollektivmitglied zum nächsten, bis der gesamte Staat informiert ist.*[23]
Eine globale Signalverbreitung ist mit lokal begrenzten Signalen demnach ebenfalls möglich. Ein technisches System muss hierzu die Fähigkeit zur gestaffelten Signalweitergabe von einer Einheit zur nächsten besitzen.

18 Vgl. HÖLLDOBLER, Bert; WILSON, Edward: The Ants. Harvard 1990, S.245

19 Vgl. JOHNSON, Steven: Emergence. New York, 2004, Seite 77.

20 KIRCHNER, Walter: Die Ameisen-Biologie und Verhalten. München, 2001, S. 72.

21 Vgl. KIRCHNER, Walter: Die Ameisen-Biologie und Verhalten. München, 2001, S. 72-73.

22 SCHMICKL, Thomas: Sammeln, Verteilen und Bewerten von Informationen. Verteilte Intelligenz in einem Bienenvolk, in: FACTS Die Informationsgesellschaft, 1, Wien 2003, S. 5-6.

23 Vgl. SCHMICKL, Thomas: Sammeln, Verteilen und Bewerten von Informationen. Verteilte Intelligenz in einem Bienenvolk, in: FACTS Die Informationsgesellschaft, 1, Wien 2003, S. 5.

3.2.4 Zusammenfassung

Die Organisationsstruktur *staatenbildender Insekten* gleicht der des in der These geforderten Systems. Die Organisationsprinzipien *staatenbildender Insekten* können daher als Vorbild für die Entwicklung eines künstlichen Systems dienen. Es konnten Prinzipien für die meisten Organisationsprobleme dezentraler Systeme gefunden werden. Die Organisation *staatenbildender Insekten* weist einige Schwachstellen im Bereich der Gefahrenabwehr auf. Diese betreffen vor allem die Bekämpfung von Eindringlingen, die sich als Kollektivmitglieder tarnen. Hiefür muss nach einer Lösung außerhalb der Prinzipien der staatenbildenden Insekten gesucht werden.

Die Kommunikationsprinzipien *staatenbildender-Insekten* können im Bereich der pheromoniellen Kommunikation übernommen werden. Hierzu ist es notwendig, einen technischen Ersatz mit pheromonähnlichen Eigenschaften zu finden. *Staatenbildende Insekten* nutzen neben der pheromoniellen Kommunikation auch den körperlichen Kontakt zur Kommunikation. Dies ist technisch schwer umsetzbar. Ob sich die Kommunikationsprinzipien, die auf taktiler Kommunikation beruhen, auf ein technisches Kommunikationsmittel übertragen lassen, bleibt zu prüfen.

Die Perfektion natürlicher Systeme kann weder im physikalischen Bereich noch im Bereich der Regelwerke technisch erreicht werden. Es muss daher geprüft werden, inwieweit sich die gefundenen Prinzipien für eine technische Umsetzung abstrahieren lassen.

3.3 Theoretische ADRRM-Systemstruktur

In diesem Kapitel wird untersucht, ob es möglich ist, auf der Basis der Prinzipien und Strategien der Metamodelle ein theoretisches System zu erstellen, das den Anforderungen der These gerecht wird.

3.3.1 Systemtheoretische Grundlagen

Die Systemstrukturen, die als Basis eines künstlichen Automatisierungs- und Steuerungskollektivs benötigt werden, sind systemtheoretisch bereits erforscht. Diese systemtheoretischen Grundlagen sollen hier kurz geschildert werden.

Verteilte-/Kollektive Intelligenz

Verteilte künstliche Intelligenz, auch *Distributed Artificial Intelligence* genannt, bedeutet die Aufteilung der Systemaufgaben auf die Gesamtheit der Systemeinheiten.[1]

Ein System, das sich ausschließlich aus dezentral organisierten Einheiten zusammensetzt, basiert gezwungenermaßen auf *Verteilter Intelligenz*. Ein auf *Verteilter Intelligenz* basierendes System, dessen Einheiten kooperativ zusammenarbeiten, bildet in seiner Gesamtheit eine *Kollektive Intelligenz*.[2]

Multi-Agenten-Systeme

Die Einheiten künstlicher Systeme mit *Kollektiver Intelligenz* bezeichnet man als *Agenten*. *Ein Agent agiert autonom, ist in der Lage mit anderen Agenten zu kooperieren und kann eigene Ziele erreichen.*[3]

Systeme, die aus einer Vielzahl von Agenten bestehen, werden dementsprechend als *Multi-Agenten-System* bezeichnet. *Multi-Agenten-Systeme bestehen aus einem Kollektiv von Agenten, die keinen Überblick über das Gesamtsystem haben und die keiner zentralen Kontrolle unterliegen. Die Daten eines Multi-Agenten-Systems sind über das System verteilt und die Verarbeitung der Daten erfolgt asynchron.*[4]

Regelkreise

Die Verhaltensweisen dieser Agenten können über Regeln gesteuert werden, die in *Regelkreisen* angelegt sind (siehe Abbildung 14).

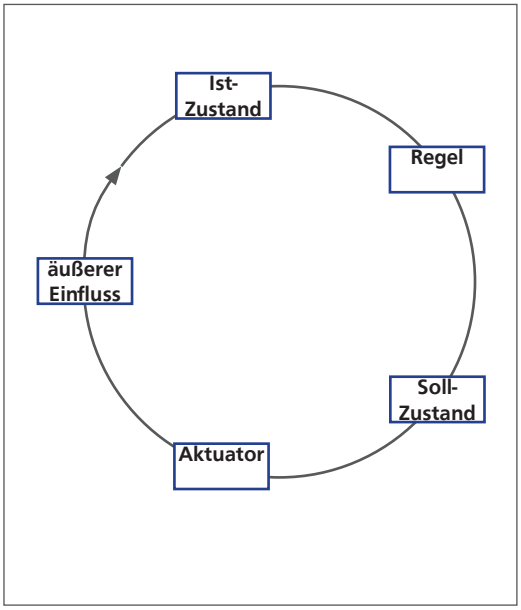


Abbildung 14:
Regelkreis
Aktuator wird betätigt bis Ist- gleich Soll-Zustand.

Regelkreise können mit *positiver Rückkopplung* oder *negativer Rückkopplung* arbeiten.

Im Falle einer *positiven Rückkopplung* wird der geregelte Vorgang durch sein Ergebnis verstärkt, im Falle einer *negativen Rückkopplung* wird der Vorgang durch sein Ergebnis gestoppt.

Positive Rückkopplungen sind nicht ungefährlich, da sie sich selbst aktivieren und nur durch externe Eingriffe gestoppt werden können. *Regelkreise mit positiver Rückkopplung* werden daher oft mit *Regelkreisen mit negativen Rückkopplung* verbunden, die nach Erreichen eines bestimmten Zustandes in den *Regelkreis* der *positiven Rückkopplung* eingreifen und ihn stoppen.

Durch die Kopplung mehrerer Regelkreise ist es möglich, sehr flexible, anpassungsfähige und dynamische Strukturen zu generieren. Eine Kopplung mehrerer Regelkreise kann sowohl innerhalb der Programmierung einer Einheit als auch durch die Kopplung mehrerer Einheiten miteinander geschehen. Agentensysteme bestehen daher aus mehreren gekoppelten Regelkreisen. Der Umgang mit solchen gekoppelten Regelkreisen ist aufgrund der Komplexität der Abhängigkeiten äußerst diffizil.

1 Vgl. pcmag.com ececlopedia: distributed intelligence. (http://www.pcmag.com/encyclopedia_term/0,2542,t=distributed+intelligence&i=41566,00.asp#), 16. Februar. 2009.

2 Vgl. SEDLACEK, Klaus-Dieter et al.: Emergenz: Strukturen der Selbstorganisation in Natur und Technik. Norderstedt 2010, S. 113-114.

3 Vgl. STARKE, Sandra: Lernen und Lernumgebung - Das Multi-Agenten-System “School Agency“. Norderstedt 2002, S. 8.

4 Vgl. SYCARA, Katia P.: Multiagent Systems, in AI Magazine 19/2. Menlo Park 1998, S. 80.

Werden die Regelkreise über Potentiale und Schwellenwerte gesteuert (siehe Abbildung 15), können sich verschiedene Ereignisse auf die gleiche Entscheidung auswirken.

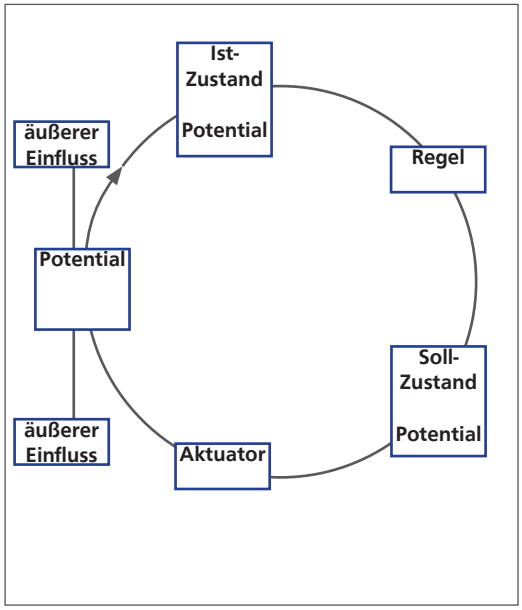


Abbildung 15:
Regelkreis
Aktuator und äußere Einflüsse wirken auf Potential

Dynamische nichtlineare Systeme

Systeme, deren Verhalten nicht allein aus augenblicklichen Ereignissen resultiert, sondern auch aus vergangenen und die daher immer identisch auf scheinbar identische Ereignisse reagieren, bezeichnet man als *dynamische Systeme.*[5] Systeme, deren Verhalten nicht proportional auf Ereignisse reagiert, werden als *nichtlineare Systeme*[6] bezeichnet.

Agentensysteme können den *dynamischen, nichtlinearen Systemen* zugeordnet werden. Vergangene Ereignisse sind in Agentensystemen beispielsweise über die Position eines Aktuators oder das Potential eines Regelkreises gespeichert. Dies kann über gekoppelte Regelkreise einen Einfluss auf aktuelle Reaktionen des Kollektivs bzw. des Gesamtsystems bewirken. Das Kollektiv reagiert dann auf ein scheinbar gleiches äußeres Ereignis nicht mehr unbedingt gleich. Damit kann es bereits als *dynamisches System* bezeichnet werden.

Des Weiteren ist es möglich, dass potentialabhängige Agentensysteme nicht linear auf äußere Ereignisse reagieren. Wurde das entscheidende Potential bereits durch andere Ereignisse beeinflusst oder wurden seine Schwellenwerte durch andere Ereignisse verschoben, erfolgt auf ein sich linear änderndes Ereignis keine sich linear ändernde Reaktion. Das Agentensystem wäre somit ein *nichtlineares System*.

Bifurkation

Ändert ein dynamisches, nichtlineares Systems unumkehrbar seine Entwicklungsrichtung, bezeichnet man dies als *Bifurkation.*[7] Ist ein *Bifurkationspunkt* erreicht, kann das System in eine Systemkatastrophe fallen. *Ein Bifurkationspunkt kann aufgrund der komplexen Abhängigkeiten in einem Agentensystem bereits durch eine kleine Veränderung der Parameter erreicht werden.*[8] Um eine Bifurkation zu vermeiden, sind verhaltensdämpfende Maßnahmen erforderlich. Diese Maßnahmen verhindern globale Auswirkungen geringer Veränderungen. Beispiele dämpfender Maßnahmen sind Filter, die geringe Werteveränderungen unterdrücken sowie ein Zurücksetzen von Schwellenwerten oder das Verrechnen mehrerer Werte zu einem Durchschnittswert.

Komplexe-Adaptive-Systeme (Complex Adaptive System - CAS)

Systeme werden in dieser Arbeit als komplex bezeichnet, wenn ihre Komponenten, Regeln und wechselseitige Kommunikation keine eindeutigen Vorhersagen über ihr Gesamtverhalten zulassen.[9] Komplexität bezeichnet eine Erscheinungsform oder ein Wahrnehmungsphänomen und keine messbare Größe. Wie und ob ein System komplex erscheint, hängt stark von dem Beobachtungsstandpunkt ab.

Vereinfacht ausgedrückt ist beispielsweise ein Ameisenstaat komplex, eine einzelne Ameise als Gesamtorganismus nicht komplex, ihr Gehirn ist komplex, ein Neuron als Ganzes nicht komplex, ein Neuron besteht aus Atomen und ist daher komplex, Atome haben einen festen Aufbau und sind somit nicht komplex. Je nach Beobachtungsstandpunkt kann ein System also eine komplexe oder eine nicht kom-

5 Vgl. MEYER, Martin: Signalverarbeitung: Analoge und digitale Signale, Systeme und Filter. Wiesbaden, 2009, S. 75.

6 Vgl. Wikipedia: Nichtlineare_Systeme. (http://de.wikipedia.org/wiki/Nichtlineare_Systeme), 25. Februar 2011.

7 Vgl. GANDOLFI, Alberto: Menschen und Ameisen. Zürich 2001, S. 74

8 Vgl. LORENZ, Edward N.: Deterministic Nonperidoc Flow, in Journal of the Atmospheric Sciences, 20, 1963, S. 130.

9 Vgl. GANDOLFI, Alberto: Menschen und Ameisen. Zürich 2001, S. 27

plexe Charakteristik aufweisen.[10]
Adaptive Systeme können sich durch Veränderung ihrer Regeln oder ihrer Komponenten an veränderte Umgebungsbedingungen anpassen. Dies kann durch selbst organisiertes Lernverhalten oder evolutionäre Mechanismen geschehen.
Der Begriff *Komplexes-Adaptives-Syste*m wird in dieser Arbeit im Sinne der Definition von John Holland verwendet:

> „Cas [complex adaptive systems] are systems that have a large numbers of components, often called agents, that interact and adapt or learn.“[11]

Emergenz

Die Definitionen des Begriffs Emergenz erweisen sich als ebenso vielfältig wie diffus. In dieser Arbeit wird der Begriff Emergenz in der folgenden Bedeutung verwendet:

Emergenz beschreibt das Phänomen globaler Systemzustände, die sich nicht direkt aus den Systembestandteilen ableiten lassen.

Diese Definition des Autors stützt sich auf die Definitionen von:

Ibrahim Abubakr:

> „Emergenz entsteht aufgrund der Wechselwirkungen von Systemelementen, die ihrerseits nicht über globale Eigenschaften des Systems verfügen.“[12]

Klaus-Dieter Sedlacek:

> „Emergenz ist die spontane Herausbildung von neuen Eigenschaften oder Strukturen auf der Makroebene eines Systems infolge des Zusammenspiels seiner Elemente. Dabei lassen sich die emergenten Eigenschaften des Systems nicht - oder jedenfalls nicht offensichtlich - auf Eigenschaften der Elemente zurückführen, die diese isoliert aufweisen. Synonyme sind Übersummativität und Fulguration.“[13]

und Steve Johnson:

> „The movement from low-level rules to higher-level sophistication is what we call emergence.“[14]

Emergenz teilt sich in zwei Kategorien, die schwache und die starke Emergenz, die nach Klaus-Dieter Sedlacek wie folgt definiert werden können:

> „Emergenz ist grundsätzlich in einer schwachen und einer starken Form denkbar. Die *schwache Form* der Emergenz entspricht einer nur vorläufigen Nichterklärbarkeit emergenter Systeme auf der Grundlage der Beschreibung ihrer Elemente. Dagegen wird bei der *starken Form* [...] auch die prinzipielle Nichterklärbarkeit angenommen.
> [...]
> Gegner der starken Emergenzthese argumentieren, dass viele ehedem als emergent erklärte Eigenschaften des menschlichen Bewusstseins sich durch die Kenntnis der Eigenschaften der Bestandteile des Gehirns (z.B. der Nervenzellen und der Synapsen) erklären ließen.“[15]

Es kann davon ausgegangen werden, dass künstliche Systeme in den Bereich der *Schwachen Emergenz* gehören. Da die Elemente künstlicher Systeme einzig auf determierbaren Strukturen basieren können, sind ihr Systemzutände grundsätzlich ebenfalls determinierbar. Somit wären künstliche Systeme grundsätzlich nicht emergent.
Aus der Definition der *schwachen Emergenz* geht indirekt hervor, dass Emergenz ein Wahrnehmungsphänomen ist; bei einer „*vorläufige Nichterklärbarkeit*“ kann keine objektive Tatsache vorliegen, es muss sich um eine Beobachtung bzw. um eine Wahrnehmung handeln. Systeme *schwacher Emergenz* können somit nur emergent *erscheinen*.
Es handelt sich bei diesen Systemen um so genannte *deterministisch chaotische*[16] Systeme. *Chaos bezeichnet in dieser Arbeit Systemzustände, die irregulär erscheinen aber nicht zwangläufig unstrukturiert*

10 Vgl. JOHNSON, Steven: Emergence. New York 2004, S. 115-116.

11 HOLLAND, John Henry: Studying Complex Adaptive Systems, in: Journal of Systems Science and Complexity, 19, Berlin 2006, S. 1.

12 ABUBAKR, Ibrahim: Ökosysteme, in: Komplexe Adaptive Systeme. Beiträge des Instituts für Umweltsystemforschung der Universität Osnabrück, MATTHIES, Michael, PAHL-WOSTL, Claudia, EBENHÖH, Eva (Hrsg.), 27, Osnabrück 2003.

13 SEDLACEK, Klaus-Dieter et al.: Emergenz: Strukturen der Selbstorganisation in Natur und Technik. Norderstedt 2010, S. 44.

14 JOHNSON, Steven: Emergence. New York 2004.

15 SEDLACEK, Klaus-Dieter et al.: Emergenz: Strukturen der Selbstorganisation in Natur und Technik. Norderstedt 2010, S. 44-45.

16 Vgl. CRAMER, Friedrich: Chaos und Ordnung. Die komplexe Struktur des Lebendigen. Frankfurt/M. 1993, S. 159.

oder ohne Ordung sind.[17] Am Beispiel der sogenannten seltsamen oder chaotischen Attraktoren ist zu erkennen, dass chaotische Systeme durchaus eine erkennbare, aber nicht erkärliche Ordnung aufweisen können (siehe Abbildung 16).

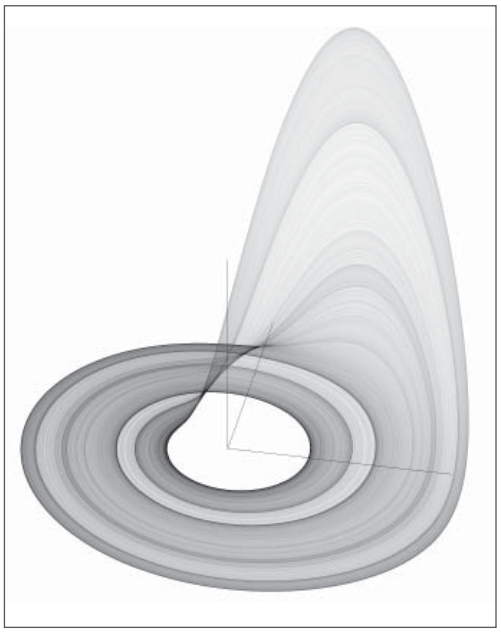


Abbildung 16:
Rössler Attraktor
seltsamer oder chaotischer Attraktor

Ein an sich determinierbares System erscheint chaotisch, wenn sich seine dem Systemzustand zu Grunde liegenden Ursachen nicht exakt wiederholen lassen. In der Folge kommt es dann zu nicht vorhersagbaren Systemverhaltenweisen und das System erscheint somit *emergent*.
Deterministisch chaotische Systeme können beispielsweise durch eine Nichtwiederholbarkeit der Ereignisse in Kombination mit *dynamischen nichtlinearen* Regeln entstehen. Kleinste Veränderungen der Ursachen, die sich der Wahrnehmung des Beobachters entziehen, können dabei zu völlig anderen Systemverhaltensweisen führen. Der Beobachter kann dann trotz seiner Kenntnis der Funktionsweise der einzelnen Einheiten und der Umgebungsereignisse keine Vorhersage des Gesamtsystemverhaltens treffen. Damit erscheint ihm das Verhalten (*schwach -*) *emergent*.
Chaotische Syteme ohne erkennbare globale Ordnung erscheinen nicht *emergent*, da *Emergenz* eine scheinbare globale Ordung voraussetzt. Ein Steuerungs- und Automatisierungssystem in Form eines dezentral organisierten *Mulit-Agenten-Systems* muss in der Lage sein, globale Aufgaben sinnvoll zu regeln. Da es sich bei diesen *Mulit-Agenten-Systemen* um *deterministisch chaotische Systeme* handeln wird, die globale Aufgaben sinnvoll regeln, werden sie zwangsläufig *emergent* erscheinen; allerdings nur in Form der *schwachen Emergenz*.
Die Fähigkeit, *emergentes Verhalten* zu entwickeln, ist die Grundvoraussetzung, die dezentrale Systeme benötigen, um globale Aufgaben zu bewältigen.

Fuzzy-Logic

Die Verwendung von Regelkreisen, deren Entscheidungen auf Potentialen und mehreren variablen Schwellenwerten beruhen, impliziert die Verwendung weicher oder sogenannter *Fuzzy-Logik*. An einem Beispiel erklärt, bedeutet dies, dass auf einen bestimmten Helligkeitswert nicht immer die gleiche Reaktion erfolgt. Der Helligkeitswert beeinflusst das Helligkeitspotential. Dieses ist statt mit einem, mit zwei Schwellenwerten ausgestattet. Ein Schwellenwert würde bei Überschreitung der Schwellen beispielsweise den Wert 1 liefern, bei Unterschreitung den Wert 0. Im Falle der zwei Schwellen wird bei Unterschreitung des unteren Schwellenwertes der Wert 0 ausgegeben. Bei seiner Überschreitung geschieht keine Veränderung des Ausgabewertes. Erst bei einer Überschreitung des oberen Schwellenwertes kommt es zu einem Wechsel der Ausgabe zu 1. Wird nun der obere Schwellenwert unterschritten, geschieht wiederum kein Wechseln des Ausgabewertes, obgleich der identische Helligkeitswert vorher eine Ausgabe von 0 zur Folge hatte. Der Bereich zwischen den Schwellenwerten ist nicht eindeutig definiert, ein Wert kann hier der Bedeutung von eher 1 als auch der Bedeutung von eher 0 entsprechen und auch die Aktuatoren könnten dementsprechende Positionen einnehmen. Dies entspricht einer weichen, einer *Fuzzy-Logik*.[18]

17 Vgl. Academic dictionaries and encyclopedias: Chaostheorie. (http://de.academic.ru/dic.nsf/dewiki/247319), 10. Juni 2011.

18 Vgl. BRÄUER, Holm: Fuzzy Logic und Wahrscheinlichkeit. (http://tu-dresden.de/die_tu_dresden/fakultaeten/philosophische_fakultaet/iph/thph/braeuer/lehre/putnam_bedeutung/Fuzzy%20Logic.pdf), Dresden 2006, S. 1.

Systemtheoretische Einordnung von ADRRM-Systemen.

Die systemtheoretischen Aspekte können in dieser Arbeit nur angerissen werden. Es kann hier einzig um die systemtheoretische Einordnung von Systemen gehen, die die Anforderungen der These nach dem ADRRM-Prinzip erfüllen.

Der These gemäß wird es sich um dezentrale Systeme ohne adressierbare Einheiten handeln. Diese Einheiten werden in künstlichen Systemen als *Agenten* bezeichnet. Systeme, die sich aus mehreren *Agenten* zusammensetzen, werden als *Multi-Agenten-Systeme* bezeichnet. ADRRM-Systeme sind folglich *Multi Agenten-Systeme*.
Die Agenten werden *Fuzzy-Logik-Elemente* zur Entscheidungsfindung benötigen, um flexibel auf Ereignisse reagieren zu können. Die Verhaltensweisen der Agenten werden über die Schwellenwerte der *Fuzzy-Logik-Elemente* gesteuert. Die Modifikation der Verhaltensweisen erfolgt über eine evolutionäre Veränderung ebendieser Schwellenwerte.
Diese *dynamischen nichtlinearen Multi-Agenten-Systeme,* deren *Fuzzy-Logik-Elemente* über Schwellenwerte zur Entscheidungsfindung verfügen, entwickeln mit großer Wahrscheinlichkeit nicht vorhersagbare Verhaltenweisen. Das System wird somit *komplex* erscheinen und seine Verhaltensweisen werden auf den Beobachter *emergent* wirken.
Aufgrund ihrer selbstanpassenden Fähigkeiten in Kombination mit einem komplex erscheinenden Systemverhalten lassen sich ADRRM-Systeme den *Komplexen Adaptiven Systemen* zuordnen.
In dezentralen Systemen müssen die globalen Aufgaben auf die einzelnen Systemeinheiten verteilt werden, die Systeme werden daher *Kollektive Intelligenz aufweisen*.

ADRRM-Systeme werden also *komplex* erscheinende *dynamische nichtlineare Multi-Agenten-Systeme* werden, die eine *Kollektive Intelligenz* aufweisen. Die Kollektive ihrer durch *Fuzzy-Logik-Regelkreise* gesteuerten Agenten werden dabei *emergente* Verhaltensweisen entwickeln.

3.3.2 Systemstruktur

In diesem Abschnitt wird die Umsetzung der Prinzipien und Strategien der Metamodelle in eine theoretische Systemstruktur dargestellt.

Kommunikation über Minimalinformationen

ADRRM-Systeme kommunizieren auf Basis minimaler Informationen. Dies bedeutet, es werden weder Handlungsanweisungen noch Sensordaten übertragen. Der Informationsaustausch erfolgt einzig auf der Basis eines Satzes allgemein gehaltener Signale, deren Bedeutung in dem ADRRM-Protokoll vorgegeben ist. Diese Signale beinhalten rein qualitative Angaben. Diese Angaben beinhalten beispielsweise die Aussage: *Umgebungshelligkeit im Tageslichtbereich*. Die Sensorwerte werden direkt von den Einheiten ausgewertet und dann in allgemeiner Form an die anderen Einheiten weitergegeben. Die Reaktion auf die empfangenen Signale obliegt den einzelnen Einheiten, sie heben oder senken daraufhin ihre entsprechenden Potentiale.

Die Kommunikation auf Basis eines Satzes allgemein gehaltener Signale bietet zwei Vorteile.

Zum Ersten sichert die Verwendung identischer Signale durch alle Einheiten das System vor einem externen Abhörangriff. Der Angreifer empfängt eine große Anzahl identischer Signale, die er keiner spezifischen Einheit zuordnen kann. Hört er Systeme ab, die konkrete Werte übermitteln, lassen sich Systemeinheiten anhand spezifischer Werte identifizieren. Zum Zweiten verringert sich das Datenvolumen erheblich: Einerseits werden Daten nur dann gesendet, wenn sie relevante Werte erreichen. Andererseits würde die Übermittlung konkreter Sensorwerte eine Information über den Wertetyp und den konkreten Wert erfordern. Dies würde mindestens ein Byte für die Typisierung und ein bis vier Byte für den Wert erfordern. Die Kommunikation über allgemeine Signale erfordert nur ein bis zwei Datenbyte.

Dezentrale Systemstruktur

ADRRM-Systeme bestehen aus einem Kollektiv kooperierender *Hardwareagenten*. Dieses Kollektiv ist dezentral organisiert und bildet ein *komplexes Multi-Agenten System* (siehe Abbildung 17).

In ADRRM-Systemen exisitieren grundsätzlich keine zentralen Elemente. Zentrale Elemente stellen grundsätzlich Angriffspunkte für externe Angriffe dar, da hier über einen Punkt ein Zugriff auf das gesamte System möglich ist. Es existieren daher weder zentrale Programmierschnittstellen noch eine zentrale Systemüberwachung.

Die Systemstruktur ist in jeder Hinsicht dezentral ausgelegt.

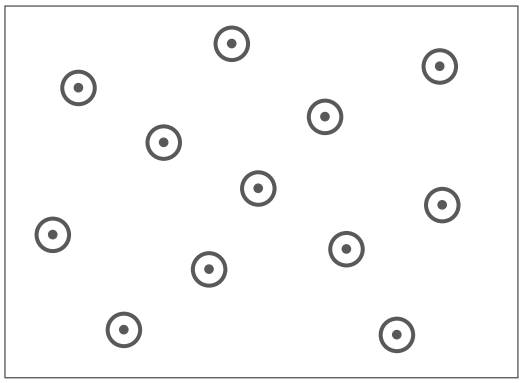


Abbildung 17:
Dezentrale Struktur

Anonyme adresslose Einheiten

Mit Adressen versehene Systemeinheiten lassen sich bei einem externen Abhörangriff bzw. Systemscan identifizieren. Anhand der dabei gewonnenen Daten ist es einem Angreifer möglich, eine Systemstruktur zu interpolieren. Um dem vorzubeugen, müssen ADRRM-Systeme ohne identifizierbare Einheiten arbeiten. Folglich besitzen ADRRM-Systeme keine adressierbaren Einheiten. Rein softwarebasierte Netzwerke können nicht ohne adressierbare Einheiten operieren, da die Informationsverbreitung innerhalb des Netzwerkes nicht kontrollierbar wäre. Dies hätte zur Folge, dass sich sämtliche Informationen über das gesamte Netzwerk verbreiten. Eine sinnvolle Arbeit wäre dem System nicht mehr möglich.

ADRRM-Systeme müssen daher über einen alternativen Mechanismus zur adressbasierten Beschränkung der Informationsausbreitung verfügen. Da ADRRM-Systeme ausschließlich für die Steuerung von Realräumen entwickelt werden, können sie sich dies zunutze machen. Die Beschränkung der Informationsausbreitung muss in ADRRM-Systemen nicht über die Software erfolgen, sie kann ebenso physikalisch begrenzt werden. Die physikalische Begrenzung richtet sich nach dem Vorbild der Pheromone der staatenbildenden Insekten. Pheromone sind flüchtige chemische Botenstoffe, deren Wirkung durch die Abnahme der Stoffkonzentration zeitlich und räumlich beschränkt ist.

Ein pheromonähnliches künstliches Signal erreicht ausschließlich Einheiten, die sich innerhalb des Wirkungsradius befinden, in dem die Signalkonzentration hoch genug ist. Einheiten außerhalb des Wirkungsradius können das Signal nicht wahrnehmen [siehe Abbildung 18].
Die Informationsausbreitung ist auf diese Weise ohne Adressierung der Einheiten begrenzbar.

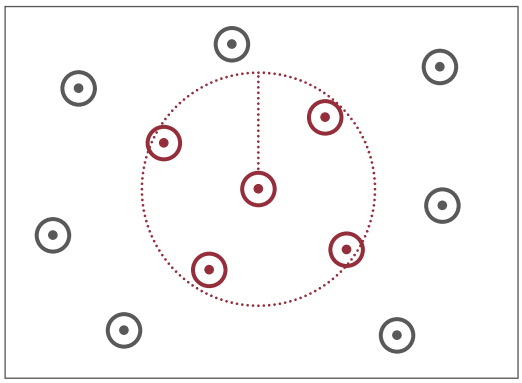


Abbildung 18:
Ausbreitungsbegrenzung durch Reichweitenbeschränkung

Kontrollierte Informationsverbreitung

In einigen Fällen kann eine Informationsverbreitung über den unmittelbaren Wirkungsradius einer Einheit hinaus erforderlich sein. Dies erfordert einen eigenen Mechanismus. Am Beispiel der Ameisen und Bienen war ersichtlich, dass Informationsweitergaben von einer Einheit zur anderen eine räumlich erweiterte, aber dennoch begrenzte Informationsverbreitung leisten kann. Auch in einem künstlichen System kann dieser Mechanismus umgesetzt werden. Hierzu muss den Einheiten entweder bekannt sein, welche Informationstypen sie weiterleiten sollen, oder sie erhalten Informationen mit einem angehängten Weiterleitungsfaktor (siehe Abbildung 19).

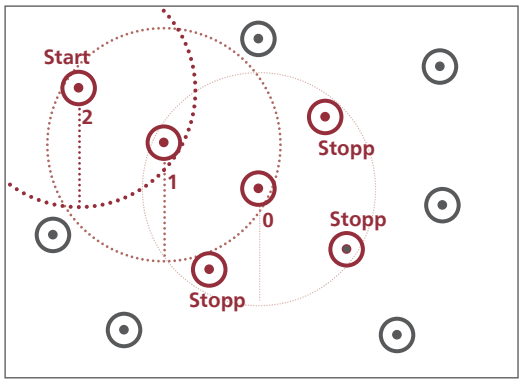


Abbildung 19:
Signalweiterleitung: Schritte beschränkt

Dieser Faktor gibt an, ob und wie oft eine Information weitergeleitet werden soll. Nach jeder Weiterleitung wird der Faktor reduziert, bis er auf Null gesunken ist und die Siganlweiterleitung stoppt.

Sollte eine globale Informationsverbreitung notwendig sein, beispielsweise im Gefahrenfall, kann der Weiterleitungsfaktor sehr hoch eingestellt werden oder auf einen Wert, der den Einheiten eine ungebremste Weiterleitung signalisiert (siehe Abbildungen 20, 21, 22).

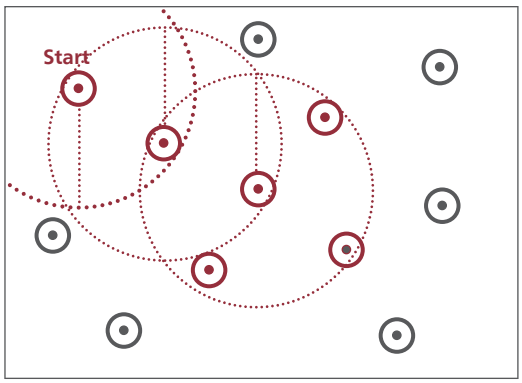


Abbildung 20:
Signalweiterleitung: Schritte unbeschränkt - Teil 1

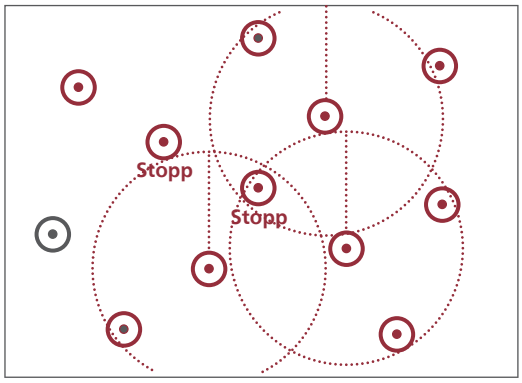


Abbildung 21:
Signalweiterleitung: Schritte unbeschränkt - Teil 2

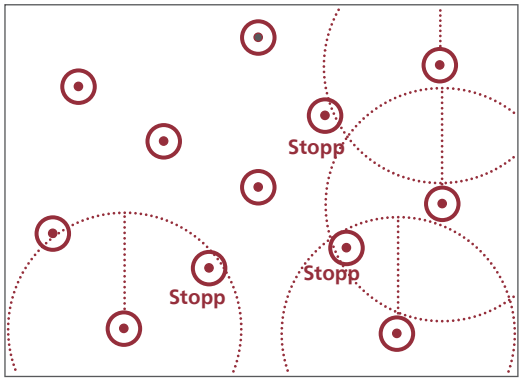


Abbildung 22:
Signalweiterleitung: Schritte unbeschränkt - Teil 3

Weiterleitbare Signale können allerdings selbstaktivierende Auswirkungen generieren. Um eine permanente Selbstaktivierung durch rundlaufende Signale zu vermeiden, muss ein weiterer Mechanismus eingefügt werden.
Ein Prinzip zu Vermeidung rundlaufender Signale findet sich bei Neuronen. Nachdem ein Neuron ein Signal übermittelt hat, benötigt es eine bestimmte Zeit, bis es wieder bereit ist ein neues Signal zu senden, diese Pause wird als *Refraktärzeit*[1] bezeichnet. Eine Pause nach dem Senden eines Signals könnte demnach eine Maßnahme zu Vermeidung einer Selbstaktivierung sein.
Ein anderer Mechanismus ohne natürliches Vorbild besteht in einer Regel, die das zweimalige Senden der gleichen Information direkt hintereinander untersagt.
Eine oder eventuell beide Mechanismen zu Vermeidung selbstaktivierender Signale sollte bei ADRRM-Systemen zum Einsatz kommen.

Adressloses Kommunikationsprotokoll

Kommunikationsprotokolle werden nach dem OSI -Modell (Open Systems Interconnection Reference Model[2]) strukturiert (siehe Abbildung 24).

Tabelle 1.1 Das OSI-Modell in der Übersicht. Jede Zeile beschreibt einen Layer des Modells.

Layer VII	Anwendungsschicht	(Application)
Layer VI	Darstellungsschicht	(Presentation)
Layer V	Kommunikationsschicht	(Session)
Layer IV	Transportschicht	(Transport)
Layer III	Vermittlungsschicht	(Network)
Layer II	Sicherungsschicht	(Data Link)
Layer I	Physikalische Schicht	(Physical)

Abbildung 24:
OSI Modell - rote Layer exisitieren im ADRRM nicht

Nach dem OSI-Modell übernehmen in dem ADRRM-Protokoll die Transceiver die physikalische Schicht, den *physical layer*. Der *physical layer* bezeichnet den technischen Teil des Systems, der für die physikalische Signalübertragung zuständig ist.
Die Transceiver übernehmen ebenso die Sicherungsschicht den *data link layer*. Diese Schicht stellt die Datenübertragung sicher. Dies geschieht, vereinfacht dargestellt, indem der Empfänger dem Sender mitteilt, ob er die Daten vollständig erhalten hat. Dieses Verfahren ist grundsätzlich auch in adresslosen Systemen einsetzbar, kann aber durch Redundanzen ersetzt werden.

Die Vermittlungsschicht, der *network layer*, der den Datentransport durch das Netzwerk regelt, existiert in ADRRM-Systemen ebenfalls nicht. Die Aufgaben des *network layer* werden durch die physikalische Reichweitenbeschränkung übernommen.

Die Transportschicht, der *transport layer*, ist ebenso inexistent. In ADRRM-Systemen werden keine zusammenhängenden Datenpakete erzeugt, deren Übermittlung geregelt werden müsste. Eine Kollisions- bzw. Signalüberschneidungskontrolle ist in ADRRM-Systemen aufgrund der redundanten Signalübertragung nicht notwendig.

Die Sitzungsschicht, der *session layer,* entfällt gänzlich. Einzelnen Sitzungen, die eröffnet und geschlossen werden, existieren in ADRRM-Systemen nicht.

Die Darstellungsschicht, der *presentation layer,* ist unnötig. Da in ADRRM-Systemen kein Datenaustausch mit externen Systemen stattfindet, wird eine Umwandlung der Daten in standardisierte Formate nicht benötigt.

Auch die Anwendungsschicht, der *application layer,* wird nicht benötigt, da es keine Anwendungen gibt, aus denen ausgewählt werden könnte.

Das ADRRM-Protokoll wurde extrem reduziert. Die Kommunikation erfolgt auf Basis von Signalwörtern, ohne Adressierung und ohne die meisten normalerweise gebräuchlichen Protokollschichten (siehe Abbildung 24). Signalwörter werden im ADRRM Protkoll zu einem beliebigen Zeitpunkt ohne Kollisionskontrolle gesendet. Der Erfolg der Datenübermittlung wird nicht kontrolliert. Die Signale werden von allen im Wirkungsradius befindlichen Einheiten empfangen. Um ein Minimum an Übertragungssicherheit zu gewährleisten, werden die Signale pro Übertragung mehrfach gesendet. Desweiteren werden die Übertragungen bis zur Aufhebung der auslösenden Faktoren fortgesetzt.

1 Vgl. KLINKE, Rainer; BAUMANN Rosemarie: Physiologie. Stuttgart 1994, S. 69.

2 Vgl. SCHREINER, Rüdiger: Computernetzwerke: von den Grundlagen zur Funktion und Anwendung. München 2009, S. 4-7.

Kommunikationssicherheit

Das ADRRM-Protokoll erlaubt keine zeitliche Synchronisation des Funkverkehrs. ADRRM-Einheiten können somit zu einem beliebigen Zeitpunkt senden. In der Folge können Signalüberschneidungen nicht ausgeschlossen werden. Signale, die sich überschneiden, sind für den Empfänger unverständlich. Da eine Signalüberschneidung nicht ausgeschlossen werden kann, bleibt einzig der Versuch, die Wahrscheinlichkeit, mit der sie auftreten, zu verringern. Um dies zu tun, können mehrere Strategien angewandt werden.

Generell gilt, je weniger gesendet wird und je kürzer die gesendeten Signale sind, desto unwahrscheinlicher ist eine Signalüberschneidung. Des Weiteren gilt, je häufiger ein Signal gesendet wird, desto wahrscheinlicher ist es, dass eines korrekt empfangen wird. Zudem sinkt die Wahrscheinlichkeit einer Signalüberschneidung, wenn die Einheiten nicht synchron getaktet sind (siehe Abbildungen 25, 26, 27).

Die Signallänge wurde bereits durch das Prinzip der Kommunikation über Minimalinformationen auf das geringst mögliche Maß gekürzt.

Das System sendet selten Daten, da die Auswertung der Sensordaten direkt auf den Einheiten stattfindet. Eine Übermittlung der Sensordaten ist daher nicht notwendig. Daten werden außer während des Evolutionsmodus einzig zum Zweck der Kooperation gesendet. Dies ist im Vergleich zu so genannten Sensornetzwerken relativ selten der Fall.

Die redundante Signalübermittlung steht in gewissem Widerspruch zu der Forderung möglichst selten zu senden. Redundante Daten sollten daher sehr schnell hintereinander gesendet werden, und sie sollten möglichst selten übermittelt werden. Es muss eine Balance zwischen der Anzahl der redundanten Daten und der Häufigkeit, mit der sie auftreten, gefunden werden.

Für Systeme ohne synchronisierten Datenverkehr ist eine synchrone Taktung geradezu schädlich. In einem Kollektiv gleich getakteter Einheiten steigt die Wahrscheinlichkeit, dass mehrere Einheiten gleichzeitig senden. Tun sie dies, überschneiden sich ihre Signale und werden nicht korrekt oder nur als ein Signal empfangen. Diese Problematik ist auch in natürlichen Systemen zu beobachten. *So gehen Studien davon aus, dass Parkinson auf einer Synchronisation bestimmter Gehirnbereiche beruht. Diese Synchronisation kann mit einem sogenannten Hirnschrittmacher durch einen desynchronisierenden Stromstoß aufgehoben werden.*[3] Zudem neigen Netzwerke zur Eigensynchronisation.[4]

Um eine Systemsynchronisation zu verhindern, müssen daher desynchronisierende Mechanismen vorgesehen werden.

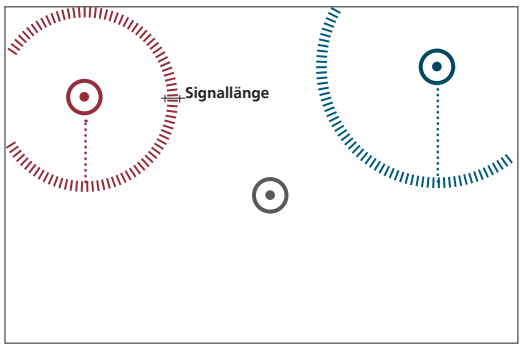


Abbildung 25:
Vermeidung von Signalüberschneidungen durch kurze Signale und asynchrones Senden - Teil 1

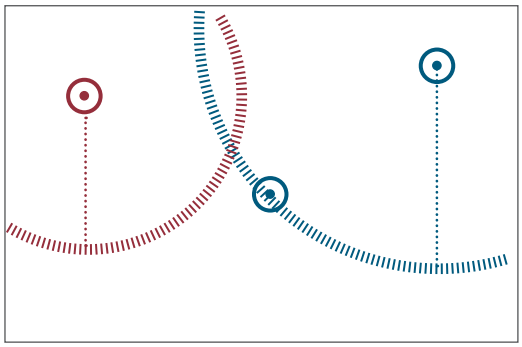


Abbildung 26:
Vermeidung von Signalüberschneidungen durch kurze Signale und asynchrones Senden - Teil 2

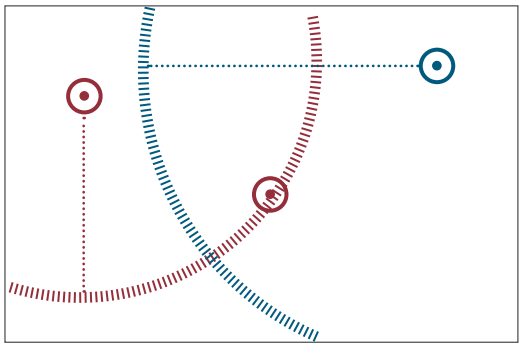


Abbildung 27:
Vermeidung von Signalüberschneidungen durch kurze Signale und asynchrones Senden - Teil 3

3 Vgl. TASS. Peter A.: Hilfe bei Parkinson und Bewegungsstörungen. Jülich 2006, S. 1.

4 METZLER, R. et al.: Synchronisation neuronaler Netzwerke, in: Physical Review E, 62, 2000, S. 2555-2565.

Kollektive Intelligenz[5]

Aktuelle Steuerungssysteme im Architekturbereich sind nach Quellenlage ausschließlich zentral strukturiert (siehe auch Kapitel 2 Stand der Technik). Ihre zentrale und hierarchische Struktur wird in der Regel durch eine zentrale *Steuerungsintelligenz* gesteuert. Diese zentrale Steuerungsstelle sammelt alle Daten, die im Steuerungsnetzwerk erfasst werden, und wertet diese in einem Umgebungssimulationsmodell aus. Dieses Modell ist zwangsläufig relativ aufwendig, da es die zu steuernden Strukturen und die erfassten Sensordaten in einer Simulation zusammenführen muss. Aus dieser Simulation kann das System Vorhersagen berechnen, aufgrund derer es entscheidet, welche Verhaltenweisen angebracht sind. Die Simulationsmodelle treffen ihre Vorhersagen aufgrund regelbasierter Umgebungsmodelle, die je nach Detaillierungsgrad extrem aufwendig ausfallen. Diese virtuellen Modelle benötigen dementsprechend leistungsfähige Rechner. Eine Änderung dieser Modelle ist aufgrund ihres Umfangs sehr aufwendig.

Werden die Steuerungsaufgaben auf ein Kollektiv verteilt, dessen Einheiten direkt in der zu steuernden Umgebung arbeiten, wird ein Steuerungsmodell obsolet. Dies ist zudem mit einem starken Rückgang des Datenverkehrs verbunden, da der Datentransport zwischen Umgebung und zentraler Steuerung entfällt. In einem dezentralen System steigt zudem die Systemrechenkapazität analog zur Systemgröße. Hierbei bleibt die Rechenbelastung der einzelnen Einheiten weitgehend stabil, da die Anzahl der unmittelbaren Nachbarn pro Einheit kaum oder gar nicht steigt.

Die Verteilung der Aufgaben einer zentralen *Steuerungsintelligenz* auf eine *Kollektive Intelligenz* ist keine triviale Aufgabe. Die linearen Steuerungsprozesse zentraler Systeme müssen zur Verteilung auf ein Kollektiv in Regeln umgesetzt werden, die in parallel arbeitenden dezentralen Systemen funktionieren.

5 Es ist dem Autor durchaus bewusst, dass der Begriff *Intelligenz* ausschließlich auf Lebewesen, je nach Definition sogar ausschließlich auf Menschen anwendbar ist. Es wäre daher in der Tat sinnvoller, im technischen Bereich einen anderen Ausdruck zu erfinden. Allerdings sind die Begriffe *Künstliche Intelligenz, Kollektive Intelligenz, Steuerungsintelligenz* und *Verteilte Intelligenz* bereits soweit etabliert, dass sie in dieser Arbeit zum besseren Verständnis ebenfalls verwendet werden.

Regelbasierte Verteilte Intelligenz

Zentrale Systeme arbeiten hierarchisch von oben nach unten, *top-down* (siehe Abbildung 28). Das globale Verhalten ist in diesen Systemen auch global gesteuert.

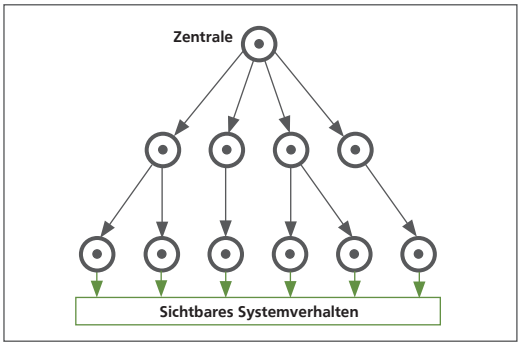


Abbildung 28:
Top-down/zentrale Systemorganisation

Dezentrale Systeme arbeiten genau umgekehrt von unten nach oben, *bottom-up* (siehe Abbildung 29). In diesen Systemen entsteht globales Verhalten aufgrund eines Zusammenwirkens einer Vielzahl lokaler Verhaltensweisen.

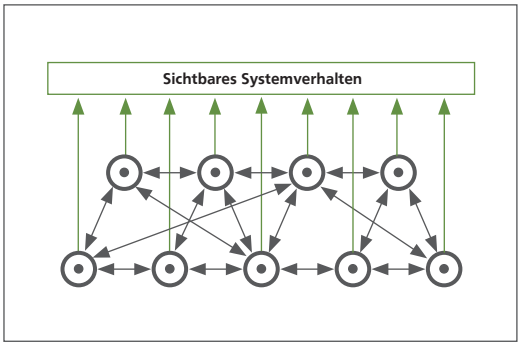


Abbildung 29:
Bottom-up/dezentrale Systemorganisation

Die strukturellen Unterschiede legen nahe, dass die Steuerung in *top-down-Systemen* völlig anders organisiert werden muss als die der *bottom-up-Systeme*.

Zur Generierung der Regeln in *bottom-up-Systemen* muss von einem lokalen Standpunkt ausgegangen werden. Das bedeutet:

Eine Einheit hat eine sehr begrenzte lokale Wahrnehmung. Die Regeln, nach denen sie handelt, basieren auf ihrer lokalen Wahrnehmung. Sämtliche anderen Einheiten unterliegen den gleichen Beschränkun-

gen.[6] Zur Regelfindung wird ein Handlungsziel festgelegt und die Abhängigkeiten, die zur Ausführung der Handlung führen sollen, werden definiert. Die Abhängigkeiten bestehen aus Umgebungsfaktoren, dem Einfluss anderer Einheiten, dem Einfluss vorhergehender Ereignisse und dem Einfluss anderer Regeln etc. und den Schwellenwerten, die festlegen ‚ab welchem Grad des Einflusses gehandelt werden soll. Die Zusammenfassung des Handlungsziels und der Abhängigkeiten ergibt die Regel.
Bei der Aufstellung einer Regel gilt es zu prüfen, ob es zu einem sinnvollen globalen Verhalten führen kann, wenn alle Einheiten nach dieser Regel handeln. Sollte das Verhalten an sich sinnvoll sein, müssen aber nicht in jedem Fall zusätzliche Abhängigkeiten in die Regel aufgenommen werden.

Die Regeln müssen zu einem weitgehend stabilen Systemverhalten führen, um den Nutzern einen sinnvollen Umgang mit dem System zu ermöglichen. Die Veränderungen des Systemverhaltens müssen so langsam verlaufen, dass die Nutzer die Chance haben, sich an die Veränderungen zu gewöhnen.
Regeln sind gut geeignet, um schnelle Reaktionen zu generieren. Die Generierung langfristiger regelbasierter Veränderungen wird als Lernfähigkeit bezeichnet.

Da sich die Umgebungsbedingungen auch grundlegend ändern können, müssen ADRRM-Systeme adaptiv ausgelegt werden. ADRRM-Systeme werden *Komplexe Adaptive Systeme* sein. Die Adaption der Systeme kann entweder manuell geschehen, was in Systemen ohne zentrale Programmierschnittstelle äußerst aufwendig wäre, oder über Selbstanpassungsmechanismen. Hiervon gibt es zwei:
Regelmodifikation über Lernfähigkeit oder über evolutionäre Entwicklung. Die Anpassung über evolutionäre Entwicklung wird in ADRRM-Systemen aufgrund der geringeren Ressourcenansprüche und der universelleren Anwendbarkeit bevorzugt (siehe auch Kapitel 3.3.3 Organisationsstrategien staatenbildender Insekten).

Regelkreise

Wie unter 3.3.1 Systemtheoretische Grundlagen bereits erwähnt, sind Regeln anhand von Regelkreisen darstellbar. Regelkreise beschreiben die Regel mitsamt ihrer Abhängigkeiten und unmittelbaren Auswirkungen.

Ein Regelkreis besteht aus äußeren Einflüssen, Sensorwerten oder Kommunikationssignalen, die das Potential des Regelkreises heben oder senken, einem Entscheidungselement, der *Weiche*, und einem Handlungselement, dem *Aktuator* (siehe Abbildung 30).

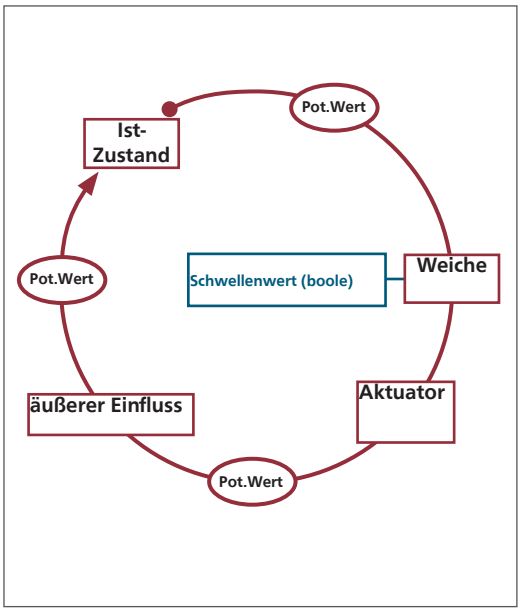


Abbildung 30:
Regelkreis mit harter Weiche

Der Wert des Potentials wird an der Weiche ausgewertet. Über- oder unterschreitet das Potential einen bestimmten Wert, Schwellenwert genannt, wird eine entsprechende Aktion ausgeführt.

Das Verhalten des Regelkreises hängt einzig von den potentialverändernden Einflüssen und den Schwellenwerten ab. Da an der Weiche in diesem Regelkreis eine boolesche (richtig/falsch) Entscheidung getroffen wird, können bereits geringe Veränderungen der Potentialwerte zu einer Verhaltensänderung führen. Diese kann wiederum gravierende Auswirkungen auf das Gesamtsystemverhalten haben. Im ungünstigsten Fall führt dies zu einer Hyperaktvität oder einem Stillstand des Systems. Um diese Systemkata-

6 In dieser Schilderung wird zunächst von ähnlichen Einheiten ausgegangen, auch wenn dies nicht zwangsläufig der Fall sein muss.

strophen zu verhindern, müssen die Schwellenwerte über *weiche Schwellen* und damit eine *weiche Logik* verfügen, die bereits erklärte *Fuzzy-Logic,* (siehe Abbildung 31).

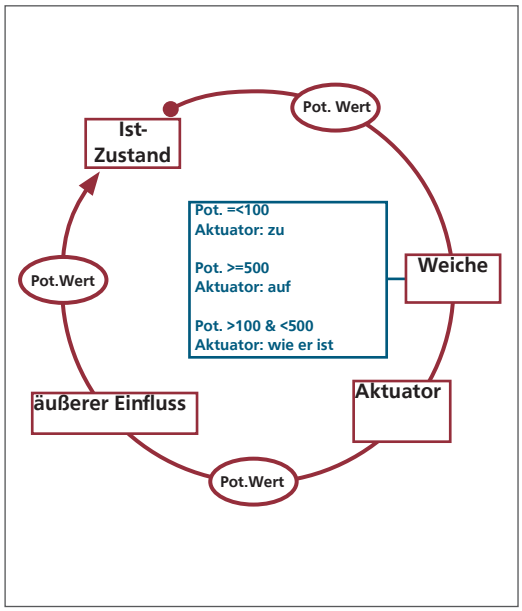


Abbildung 31:
Regelkreis mit weicher oder Fuzzy Logik

In diesem Regelkreis wurde die *boolesche Logik* durch *Fuzzy-Logik* ersetzt. Der Regelkreis besitzt zwei verschiedene Schwellenwerte, die jeweils ausschließlich bei Über- oder Unterschreitung relevant sind. Da diese Schwellenwerte in einem gewissen Abstand zueinander liegen, ergibt sich ein Bereich, in dem der Aktuator keine Positionsänderung vornimmt. In dem abgebildeten Regelkreis ist dies der Bereich zwischen 101 und 499.
Diese *Fuzzy-Regelkreise* eignen sich beispielsweise zur Steuerung von Jalousien. Es ist oft zu beobachten, dass helligkeitsabhängig gesteuerte Jalousien permanent auf- und zufahren. Dies findet seine Ursache in der *booleschen Logik* oder in zu nah beieinander liegenden Schwellenwerten. Werden die Schwellenwerte in ausreichendem Abstand zueinander eingerichtet, bewirken geringe Helligkeitsveränderungen keine Reaktion der Jalousie, da ihr Verhalten gedämpft ist.

Künstliche Neuronen

Wirken mehrere Einflüsse auf ein Potential eines Regelkreises, entspricht dieser in seiner Wirkungsweise einem *Neuron* (siehe Abbildung 32). Ein solcher Regelkreis ist in der Lage, mehrere Signale zu einem Ausgangssignal zu verarbeiten. Ist die *Weiche* des Regelkreises *boolesch* angelegt, ergibt sich ein hart schaltendes Neuron, in dieser Arbeit als *Schaltneuron (S)* bezeichnet. Besitzt der Regelkreis eine *Fuzzy -Weiche,* ergibt sich ein weich schaltendes Neuron, in dieser Arbeit als F*uzzy-Neuron (F)* bezeichnet.

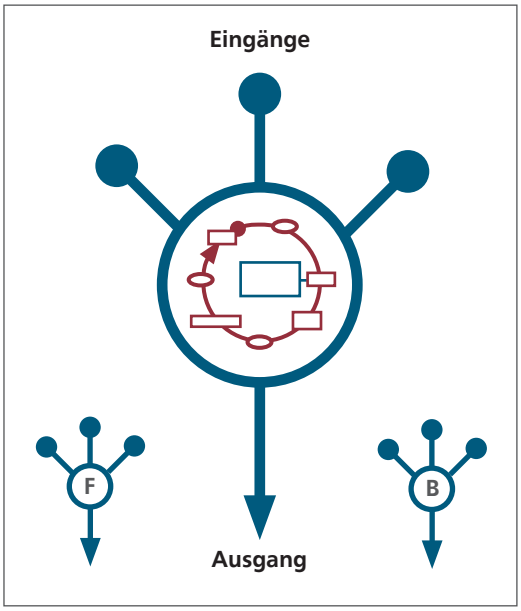


Abbildung 32:
Virtuelles Neuron mit drei Eingängen
Fuzzy-Neuron: F Boolesches Neuron: B

Obgleich es möglich wäre, aus diesen Neuronen ein neuronales Netz aufzubauen, wird diese Option in ADRRM-Systemen aus technischen Gründen[7] nicht wahrgenommen.
Die Neuronen werden in ADRRM-Systemen ausschließlich zur flexiblen Entscheidungsfindung herangezogen.

Aktivierungsebenen

Wie bereits erwähnt, arbeiten ADRRM-Systeme auf Basis einer physikalischen Begrenzung der Informationsausbreitung. Diese Begrenzung wirkt sich realräumlich aus und ist in vier Auflösungen unterteilt (siehe Abbildung 33).

Der kleinste Aktivierungsbereich beschränkt sich auf den Wahrnehmungsbereich einer Einheit. In diesem

7 Ein virtuelles neuronales Netz würde sich mit der Rechenleistung einfacher bzw. wirtschaftlich geeigneter Mikrokontrollern nicht umsetzen lassen. Es existiert zwar eine wirtschaftlich denkbare Lösung in Form von FPGAs, diese zu entwickeln ist jedoch nicht Bestandteil der Themenstellung der Arbeit und kann zu einem späteren Zeitpunkt erfolgen.

Bereich arbeitet die Einheit autonom. Der nächste Bereich umfasst die Einheiten im Signalwirkungsradius um eine Einheit. In diesem Fall kommt es bereits zu einer Kooperation mehrerer Einheiten. Der nächst größere Bereich umfasst den Signalwirkungsradius mehrerer Einheiten. Die Anzahl der kooperierenden Einheiten kann hier flexibel gehandhabt werden. In besonderen Fälle kann die Ausbreitungsbegrenzung aufgehoben werden, um das gesamte System zu aktivieren.

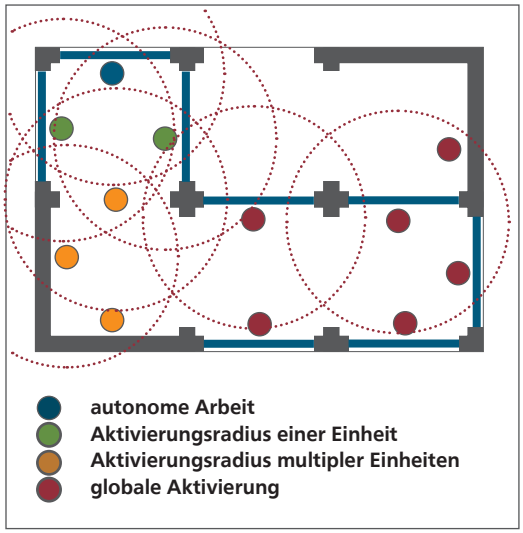


Abbildung 33:
Grundrissschema mit Aktivierungsebenen

Autonome Ebene

Die Systemeinheiten arbeiten im Ruhemodus autonom und kommunizieren nicht. Der Ruhemodus wird daher in der Regel auf der Einheitenebene betrieben.

Ebene einer Einheit (siehe Abbildung 34)

Diese Aktivierungsebene wird benutzt um lokal begrenzte Aktivitäten in einem räumlichen Umfang beispielsweise eines Arbeitsplatzes zu steuern.
Dies ermöglicht zudem die temporäre Aneignung privater Bereiche, auch wenn das ADRRM-System primär zum Einsatz in halböffentlichen und öffentlichen Räume entwickelt wurde und nicht die privaten Räume, wie sie von Smart-Home Systemen geregelt werden, abdecken möchte.

Ebene multipler Einheiten (siehe Abbildung 35)

Sollen umfangreichere Bereiche in der Größenordung eines Seminarraumes oder eines Wegeabschnittes aktiviert werden, wird diese Option gewählt.

Ebene globaler Aktivierung (siehe Abbildung36)

Diese Ebene wird hauptsächlich im Gefahrenfall genutzt, wenn auch Bereiche aktiviert werden müssen, die nicht unmittelbar von dem auslösenden Ereignis betroffen sind.

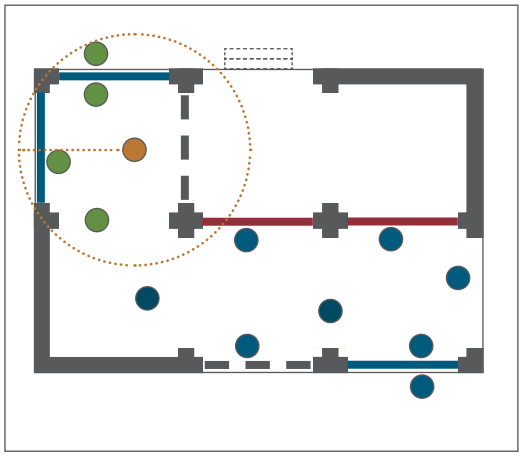


Abbildung 34:
Ebene einer Einheit

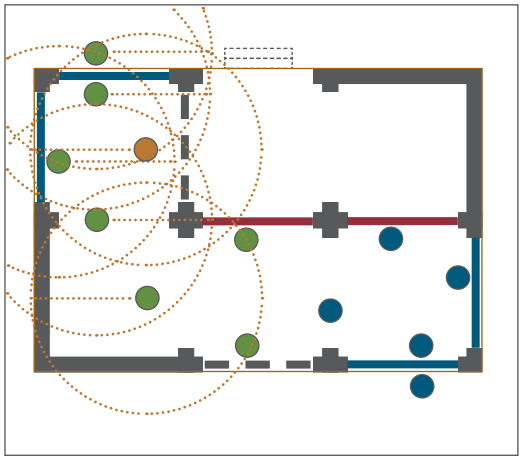


Abbildung 35:
Ebene mulitpler Einheiten

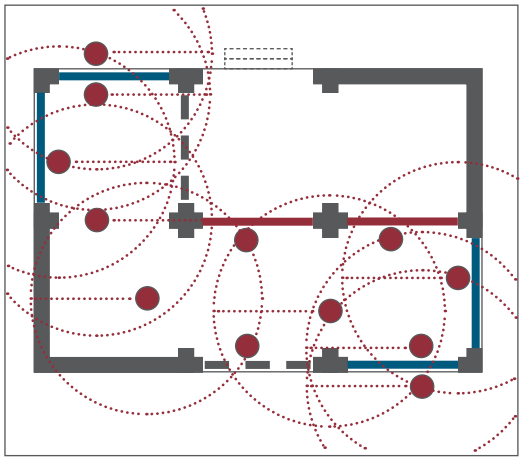


Abbildung 36:
Globale Ebene

Ebenenhierarchie

Die Ebenen sind hierarchisch von global zu autonom angelegt. Die räumlich umfangreichere Ebene hat Vorrang, da die Relevanz der Aufgaben analog den ihnen zugeordneten Eben zu bewerten ist. Das bedeutet, die Ebene, die im Gefahrenfall aktiviert wird, unterdrückt die Aktivitäten auf der Ebene der multiplen Einheiten, auf der beispielsweise die Temperaturregulierung stattfindet.
Die Kooperation, der Kommunikationsaufwand und die Aktuatoraktivitäten steigen mit zunehmender räumlicher Aktivierung an. Die Arbeit auf niedrigeren Ebene ist somit energieeffizienter. Zielvorgabe der Einheiten ist es daher, auf der räumlich am stärksten begrenzten Ebene zu arbeiten.

Evolutionäre Selbstanpassung

Lernverhalten erfolgt zielgerichtet anhand von Vorgaben und ist daher die schnellere Anpassungsmethode. Ein lernfähiges System muss allerdings in der Lage sein, die Folgen seiner Verhaltensweisen zu bewerten. Um dies zu tun, benötigt das System eine sehr detaillierte Wahrnehmung, die es ihm gestattet, die Folgen seiner Handlungen von denen fremder Handlungen zu unterscheiden. Zudem müssen lernfähige Systeme über einen Langzeitspeicher verfügen, um erfolgreiche Handlungsmuster bzw. Lernerfolge abzulegen, die sie bei passender Gelegenheit wieder aufrufen können.
Lernenfähigkeit erfordert dementsprechend umfangreiche technische Ressourcen, die im Widerspruch zu möglichst einfachen und kostengünstigen Systemeinheiten stehen.

Evolutionäre Anpassung kommt mit wesentlich geringeren Ressourcen aus. In evolutionären Systemen müssen die Einheiten ihr Verhalten nicht bewerten, ungünstige Verhaltenweisen führen schlicht zum Tod der Einheiten und damit sterben ungüstige Verhaltenweisen einfach aus. Eine feine Sensorik ist hierzu nicht notwendig und auch Zielvorgaben sind obsolet. Die Richtung, in die sich evolutionäre Systeme entwickeln sollen, wird in künstlichen Systemen von den Faktoren bestimmt, die die Einheiten zu ihrem *Überleben* benötigen. Diese Faktoren können frei gewählt werden, da der Tod der Einheiten nicht ihre physische Vernichtung meint, sondern einzig das Ende ihrer bisherigen Schwellenwerte. Die Wahl der für das Überleben einer Einheit notwendigen Faktoren wird sinnvollerweise analog zu den Faktoren ausfallen, die sie beeinflussen können.
Dies bedeutet beispielsweise, dass eine Einheit, die einen Heizkörper reguliert, in einem Bereich überleben kann, in dem eine Temperatur zischen 17°C und 25°C gehalten wird. Die Einheit hat die Aufgabe die Temperatur zwischen 19°C und 23°C zu halten. Um eventuell auftretende Messfehler herauszufiltern, sind die Toleranzwerte, die Überlebensanforderungen, etwas weiter gefasst als die optimalen Bedingungen. Ist die Einheit nicht in der Lage, die Temperatur innerhalb ihrer Toleranzwert zu halten, gilt sie als *krank*. *Kranke* Einheiten können ihre Schwellenwert nicht mehr weitergeben. *Kranke* Einheiten erhalten neue Schwellenwerte, die sie mit ihren bisherigen verrechnen. Ist dies geschehen, starten die Einheiten mit neuen Schwellenwerten und damit mit neuen Verhaltensweisen aufs Neue. Sie sind nun quasi die Nachkommen ihrer selbst und einer unbekannten Spendereinheit. Dieses Verfahren wurde gewählt, um sprunghafte Veränderungen der Schwellenwerte zu unterdrücken, die sonst zu ebenso sprunghaften Verhaltensänderungen führen würden, was den Umgang mit dem System sehr erschweren würde.

In evolutionären Systemen setzen sich die gut funktionierenden Verhaltensweisen durch. Dies kann über einen längeren Zeitraum zu einer Verarmung der Varianz der Schwellenwerte führen, zu einer Art Schwellenwertmonokultur. Dies ist systemschädigend, da dem System im Falle einer Umgebungsveränderung keine ausreichende Varianz an Schwellenwerten mehr zu Verfügung steht, um seine Verhaltenweisen in größerem Ausmaß zu verändern.
Zusätzlich existiert in evolutionären Systemen das Phänomen der *lokalen Maxima* (siehe Abbildung 38).

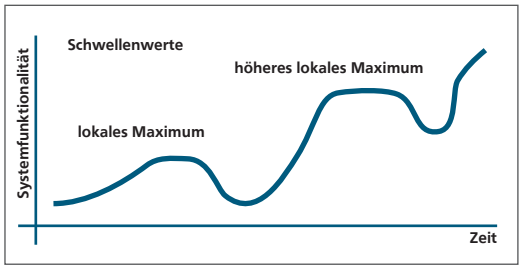


Abbildung 38:
lokale Maxima

Ein *lokales Maximum* tritt auf, wenn das System evolutionär Verhaltensweisen erreicht hat, mit denen es seine Umgebung innerhalb seiner Toleranzwerte halten kann. Es ist ein *lokales Maximum*, da es weitere Verhaltensweisen geben kann, die eine noch bessere Steuerung der Umgebung ermöglichen. Der Idealzustand wären Verhaltensweisen, die das System auf einem absoluten, einem *globalen Maximum* halten. Da das *globale Maximum* in *komplexen Systemen* nicht wahrnehmbar ist, muss das System angeregt werden, nach weiteren *lokalen Maxima* zu suchen, die eventuell höherwertig sind als das aktuelle.
Da ein einmal erreichtes *lokales Maximum* nur unter Zwang verlassen wird, denn schließlich funktioniert es ja, muss eine dynamische Entwicklung erzwungen werden. Hierzu müssen mehrere Maßnahmen kombiniert werden. Zum einen müssen permanent neue Werte erzeugt werden, die das Systemverhalten modifizieren. Dies geschieht am einfachsten durch eine Mutation der Schwellenwerte bei der Übertragung vom Spender an den Empfänger. Diese kann über eine Programmroutine geschehen und auch aufgrund von Übertragungsfehlern während der Schwellenwertübertragung. Letzteres kommt dem ADRRM-Protokoll sehr zugute, da das Protokoll die korrekte Übertragung der Daten nicht kontrolliert. Da es in funktionierenden Systemen allerdings gar nicht erst zu evolutionären Prozessen kommt, würde eine Mutation der Schwellenwerte nicht greifen, da ein Schwellenwertaustausch schlicht nicht stattfindet. Die Mutation muss daher mit einem Mechanismus ergänzt werden, der das System in einen evolutionären Prozess zwingt. Dieser Mechanismus ist aus der Natur gut bekannt, er nennt sich *Begrenzung der maximalen Lebenszeit*. Die Systemeinheiten dürfen nicht ewig leben, auch wenn ihre Verhaltensweisen gut funktionieren. Die Lebenszeit der Einheiten wird auf einen bestimmten Zeitraum begrenzt, der ebenfalls einer evolutionären Veränderung unterworfen sein kann. Ist die Lebenszeit einer Einheit abgelaufen, gilt sie als *krank* und erhält neue mutierte Schwellenwerte, die sie mit ihren vorherigen Werten verrechnen wird. Die gut funktionierenden Schwellenwerte gehen somit nicht vollständig verloren, sie existieren in modifizierter Form in der nächsten Generation weiter. Um das System vor einer Drift in eine extreme Richtung zu bewahren, können von Zeit zu Zeit die ursprünglichen Startwerte in das System eingespeist werden, dies kommt einer Art partiellem *Reset* gleich.

Evolutionäre Systeme können sich ausschließlich im Betrieb optimieren. Sie werden mit für sinnvoll angenommenen Schwellenwerten gestartet. Eine Optimierung von Anfang an ist jedoch in komplexen Systemen auszuschließen. Die Systeme werden daher eine Optimierungsphase benötigen, in der sie eine Anzahl von evolutionären Prozessen durchlaufen, bis sie das erste *lokale Maximum* erreicht haben. Um diese Phase, in der die von dem ADRRM-System gesteuerte Umgebung eventuell nicht nutzbar ist, zu verkürzen, gibt es mehrere Möglichkeiten. Die Abfolge der evolutionären Prozesse kann beschleunigt werden, so dass die Entwicklung schneller voranschreitet. Die Schwellenwerte können mit einer großen Varianz gestartet werden, so dass sich schneller gut funktionierende Werte ergeben. Eventuell kann auch eine Trainingsumgebung eingerichtet werden, um bei der Montage in die eigentliche Umgebung bereits eine Kollektiv auf einem einigermaßen passenden lokalen Maximum bereitzustellen. Diese Methode kann zwar auch anhand einer virtuellen Umgebung angewandt werden, ist jedoch mit einem sehr großen Aufwand verbunden, was ihren Einsatz eher unwahrscheinlich macht. Umgebungen, die von ADRRM-Systemen gesteuert werden, müssen wohl einige Zeit bereitgestellt werden, um dem System eine Anpassung zu ermöglichen. Allerdings passen sich ADRRM-Systeme danach permanent in nahezu unmerklich kleinen Schritten während des Betriebes an.

Abwehr externer Angriffe

Natürliche Systeme sind in der Regel physischen Angriffen ausgesetzt, dieser Gefahrenfall wird aus der Betrachtung ausgenommen, denn er würde eine physische Antwort erfordern und bevor das System jemanden verletzt, opfert man doch besser das System. Dies bedeutet allerdings, dass die Strategien staatenbildender Inseketen zur Gefahrenabwehr nicht als Vorbilder genutzt werden können.

Die Angriffe, gegen die sich ADRRM-Systeme wehren dürfen, bestehen aus Angriffen auf ihre Programmierung.

ADRRM-Systeme sind aufgrund ihrer unzugänglichen Systemstruktur vergleichsweise gut vor externen Angriffen auf die Programmierung geschützt.
Ein Eindringen über eine zentrale Schnittstelle ist nicht möglich. Selbst die Infizierung von Systemeinheiten hat keine weiterreichenden Folgen, da die Einheiten sich nicht gegenseitig umprogrammieren können. ADRRM-Einheiten nehmen einzig während des Evolutionsmodus verhaltensändernde Werte an. Während dieser Phase wäre es einem Angreifer möglich, falsche Schwellenwerte in das System einzuspeisen. Um diese Gefahr zu minimieren, ist die Zeitspanne, in der Schwellenwerte angenommen werden auf einige Sekunden beschränkt. Zudem ist der Zeitpunkt, an dem der evolutionäre Prozess stattfindet, ereignisbasiert, er ist daher nicht exakt vorhersehbar. Eine Einspeisung falscher Schwellenwerte würde allerdings während der nächsten evolutionären Phasen wieder kompensiert. Um das System vollständig gegen die Einspeisung falscher Werte zu schützen, wäre allerdings eine Verschlüsselung der Schwellenwertübertragung notwendig.
Des Weiteren können ADRRM-Systeme durch die Einspeisung falscher Sensorwerte sabotiert werden. Um dies zu vermeiden, kann entweder eine verschlüsselte Systemkennung eingeführt werden oder es werden selbstanpassende Filter eingesetzt, die außergewöhnliche Sensordaten eliminieren.

Verarbeitung multipler oder konträrer Anforderungen

Im Falle konträrer oder multipler Anforderungen müssen die Systeme in der Lage sein, Kompromisse zu finden. Hierzu stehen dem System drei Methoden zur Verfügung:

Einzelne Einheiten reagieren getrennt auf die jeweils potenteste Anforderung in ihrem direkten Umfeld. Der Regelungskompromiss erfolgt hierbei durch die unterschiedliche Aktivierung multipler Aktuatoren. Diese Methode eignet sich beispielsweise um das Öffnen der Fenster in großen Räumen zu regeln, in denen sich mehrere Nutzer aufhalten und dabei Abstände zueinander halten, die größer als die Signalwirkungsradien ihrer Einheiten sind. Hierbei würde ein Teil der Fenster geöffnet, ein anderer Teil nicht (siehe Abbildung 39).

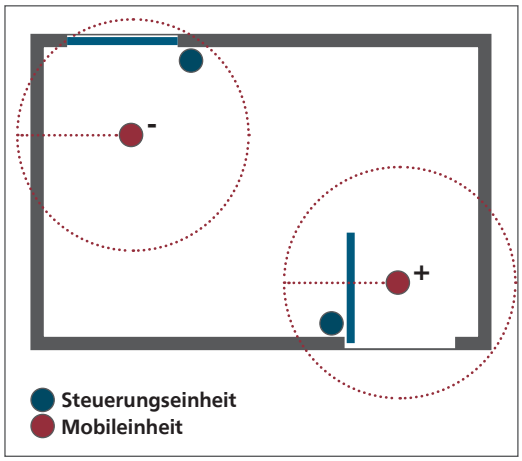


Abbildung 39:
Kompromiss - räumlich

Überschneiden sich die Signalwirkungsradien mehrerer Nutzer an einer Steuerungseinheit, könnte diese die unterschiedlichen Anforderungen miteinander verrechnen (siehe Abbildung 40). Die Mobileinheiten der Nutzer senden deren Anforderungen in einer bestimmten Taktung. Die Steuerungseinheit empfängt nun in einem bestimmten Zeitraum eine Anzahl Potential hebender und senkender Signale. Diese Signale könnte sie nun miteinander zu einem Durchschnittwert verrechnen, nach dem sie ihr Verhalten ausrichten kann. Auch hier werden multiple Akutatoren getrennt gesteuert. Diese Methode eignet sich für Räume, in denen viele Nutzer nahe zueinander positioniert sind, beispielsweise Seminarräume, Sitzungssäle oder Werkstätten.

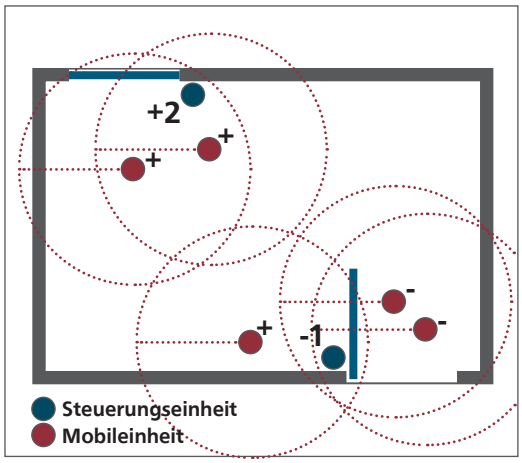


Abbildung 40:
Kompromiss - Verrechnung

In bestimmten Situationen könnte es erforderlich sein, einen allgemeingültigen Kompromiss zu finden. Dies wäre beispielsweise der Fall, wenn sich unter-

schiedliche Akutatoraktionen zueinander energieineffizient verhalten.
In diesen Fällen kann die, im Kapitel 3.3.3 Organisationsstrategien staatenbildender Insekten beschriebene Methode der dezentralen Mehrheitsbildung angewendet werden.
Hierbei wird ein mehrstufiger Abstimmungsprozess gestartet. In diesem Prozess müssen sämtliche Einheiten innerhalb eines Raumes einbezogen werden, da hier eine Kommunikation über den gesamten zu regelnden Raum vorgenommen werden muss. Die Einheiten senden nun zunächst ihre Anforderungen aus (siehe Abbildung 41).

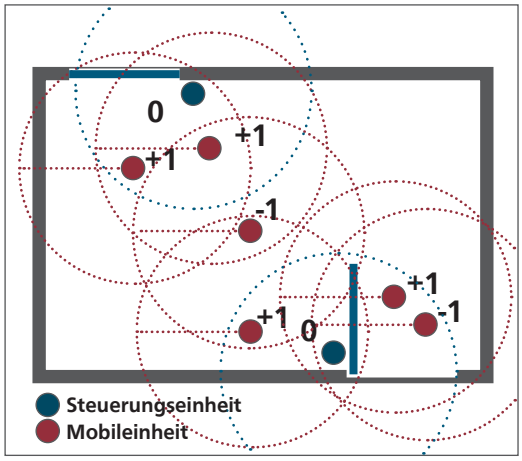


Abbildung 41:
Kompromiss - Abstimmung Teil 1

Danach empfangen sie die Anforderungen der benachbarten Einheiten und verrechnen diese zu einem Durchschnittswert, aufgrund dessen sie eine neue Anforderung definieren, die sie wiederum aussenden (siehe Abbildung 42).

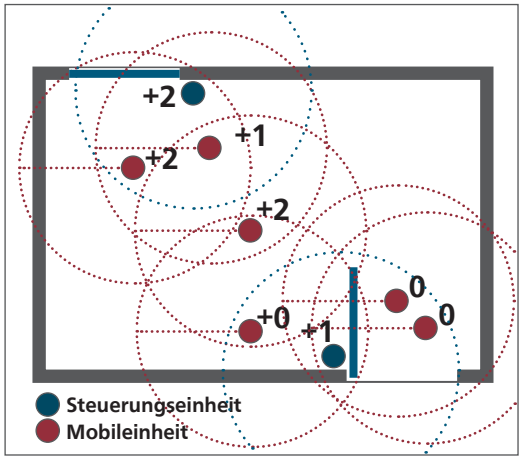


Abbildung 42:
Kompromiss - Abstimmung Teil 2

Nach einigen Wiederholungen dieses Vorgangs setzt sich eine eindeutige Anforderung durch, die von allen befolgt wird (siehe Abbildung 43).

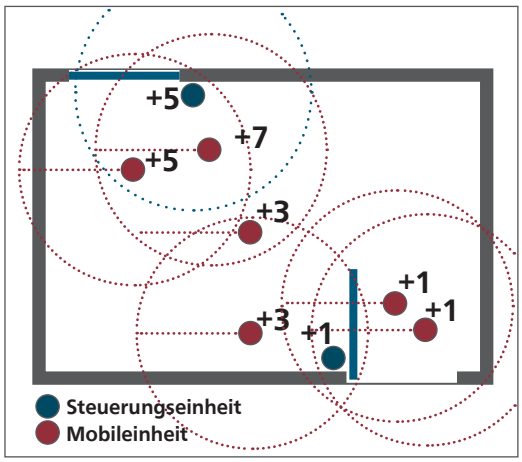


Abbildung 43:
Kompromiss - Abstimmung Teil 3

Nutzer - ADRRM-System-Schnittstellen

Über die Mobileinheit wird ein Nutzer zum Mitglied des ADRRM-Kollektivs. Er kann nun auch aktiv Anforderungen an das System stellen. Diese Forderungen beeinflussen das System innerhalb des Signalwirkungsradius seiner Einheit bzw. in einem begrenzt erweiterten Bereich (siehe auch Abschnitt: Kontrollierte Informationsverbreitung; siehe Abbildung 44).

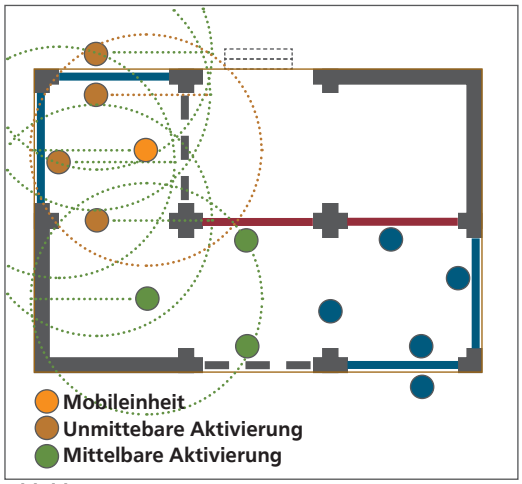


Abbildung 44:
Mobileinheit - begrenzte Einflusserweiterung

Des Weiteren können Einheiten zum Einsatz kommen, die dem Nutzer eine akustische oder optische Rückmeldung bieten. Feedbackeinheiten bieten sich vor allem zur Suche disfunktionaler Bereiche an (siehe Kapitel 3.4 Die Versuchseinheiten).

3.3.3 Systemmodi

ADRRM-Systeme arbeiten auf Basis von Betriebsmodi. Diese Modi sind maßgebend für die Systemreaktionen und unterscheiden sich vor allem in ihren Schwellenwerten.
Zunächst wird immer über den aktuell angebrachten Modus entschieden. Diese Entscheidung hängt von den Sensorwerten, den Signalen anderer Einheiten und den aktuellen Werten der Potentiale der Einheit ab. Diese werden miteinander zu neuen Potentialwerten verrechnet, die maßgebend für die Modusentscheidung sind.
Erst in einem zweiten Schritt wird über eine adäquate Handlung innerhalb des festgelegten Modus entschieden.
Die globalen Modi müssen über dezentrale Prozesse ausgelöst werden. Hierfür muss geregelt werden, wie viele Einheiten oder welche Umgebungsfaktoren zusammenwirken müssen, um das Systemverhalten global zu beeinflussen.
Soll das System beispielsweise in den Evolutionsmodus wechseln, sollten aus Redundanzgründen mehrere Einheiten beteiligt sein. Um einen Gefahrenabwehrmodus zu aktivieren, sollte bereits eine Einheit ausreichen.
Eine ereignisorientierte Regelung der Modi ist einer zeitbasierten Regelung aufgrund der größeren Flexibilität vorzuziehen. Des Weiteren wäre ein Änderung zeitbasierter Aktionen in Systemen ohne zentrale Schnittstelle äußerst aufwendig.

Die Betriebsmodi sind hierarchisch gestaffelt, die Gefahrenmodi haben Vorrang vor allen anderen (siehe Abbildung 45).

Die Auslösung eines Gefahrenmodus kann über die Sensoren der Einheit erfolgen, aber auch über den Empfang von Gefahrensignalen anderer Einheiten. Wurden keine Anzeichen für eine drohende Gefahr wahrgenommen, wird geprüft, ob die Einheit geweckt wurde. Ist dies der Fall, werden nacheinander der Appell und der Evolutionsmodus aktiviert. Ist dies geschehen, wechselt die Einheit in den Wachmodus. Fallen die Umweltfaktoren unter eine gewisse Schwelle, wechselt die Einheit in den Ruhemodus.

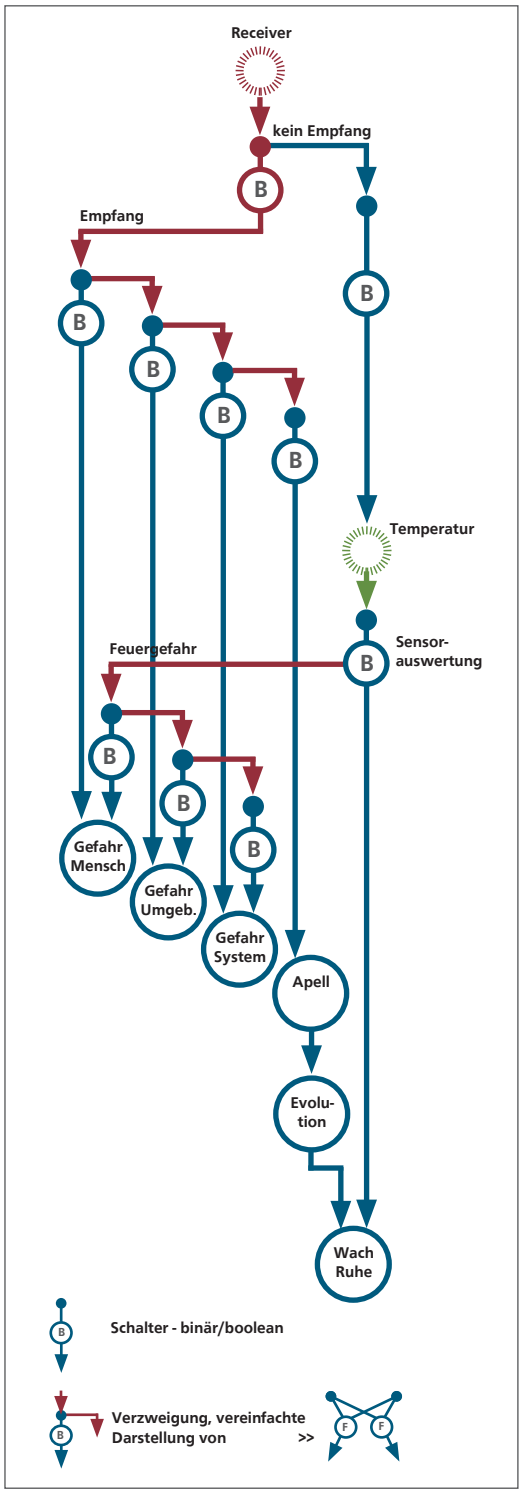


Abbildung 45:
Entscheidung über Systemmodi

Wach- und Ruhemodus

Das System benötigt im Regelbetrieb zwei globale Zustände, den Wachzustand und den Ruhezustand. Diese Zustände regeln die Systemdynamik. Welcher dieser beiden Betriebsmodi relevant ist, hängt von der Umgebungsaktivität ab. Diese kann über Temperatursensoren, Helligkeitssensoren, Passivinfrarotsensoren oder akustische Sensoren erfasst werden. Die Sensorwerte werden hierbei über *Fuzzy-Neuronen* auf ihre Relevanz geprüft, Funksignale werden direkt durchgeschaltet (siehe Abbildung 46). Steigt die Umgebungsaktivität über die Wachschwelle, erwachen die ersten Einheiten und leiten eine Weckprozedur ein.

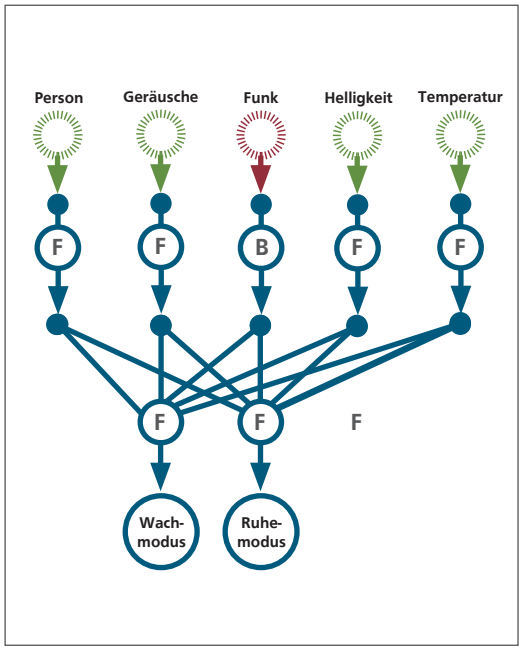


Abbildung 46:
Entscheidung Wach-/Ruhemodus

Ruhemodus

Die Reduzierung der Systemaktivitäten dient in erster Linie der Energieeffizienz. Im Ruhemodus reagiert das System träger, woraufhin weniger Aktuatoraktivitäten stattfinden.

Die Schwellenwerte der Fuzzy-Neuronen werden im Ruhemodus weit auseinander gelegt. Dies hat einen weiten Messwertbereich zur Folge, in dem die Aktuatorreaktionen weich definiert sind. Veränderungen der Umgebungsfaktoren haben somit nur im Falle extremer Veränderungen Reaktionen zur Folge.

Wechsel Ruhe- zu Wachmodus

Der Wechsel vom Ruhe- in den Wachmodus ist ein mehrstufiger Prozess, in dem vor der Aktivierung des Wachmodus noch die Systemselbstdiagnose durchgeführt wird. Da das System während dieses Prozesses keinen Steuerungsaufgaben nachkommt, sollte er zu einer wenig betriebsamen Zeit erfolgen.

Weckprozedur

Stellten die äußeren Einheiten erhöhte Umgebungsaktivitäten fest, erwachen sie und senden sich global verbreitende Wecksignale (siehe Abbildung 47). Wenn diese Signale eine bestimmte Taktung übersteigen, steigt das Wach-/Ruhepotential der Einheiten und sie verlassen den Ruhemodus (siehe Abbildung 48).

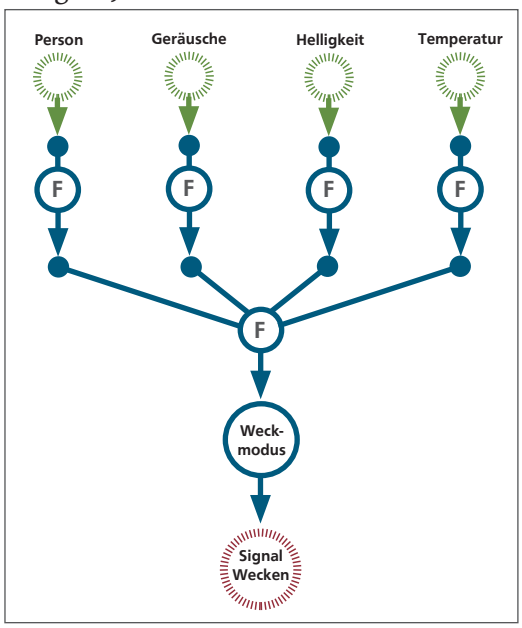


Abbildung 47:
Entscheidung Weckmodus

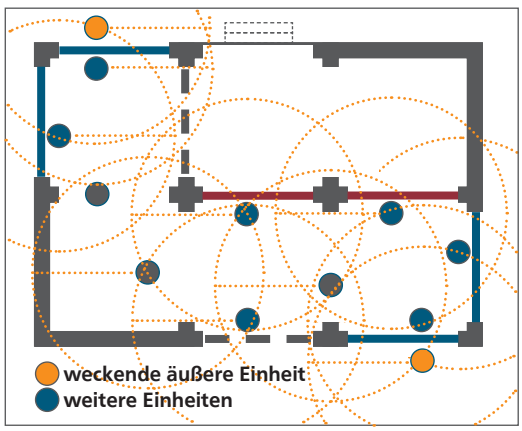


Abbildung 48:
Ausbreitung der Wecksignale

Appellmodus

Die erwachten *gesunden* Einheiten beginnen Appellsignale zu senden und signalisieren so ihre Einsatzfähigkeit (siehe Abbildung 49). Einheiten, die ihre Umgebungswerte nicht innerhalb ihrer Toleranzen halten konnten, gelten als *krank* und senden keine Appellsignale.

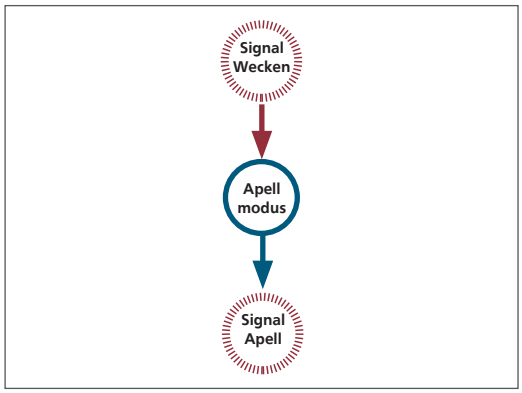


Abbildung 49:
Appellmodus 1 - Initialisierung

Die Anzahl der empfangenen Appellsignale innerhalb eines begrenzten Zeitraumes lässt Schlüsse über die Anzahl der noch *gesunden* Einheiten zu (siehe Abbildung 50).

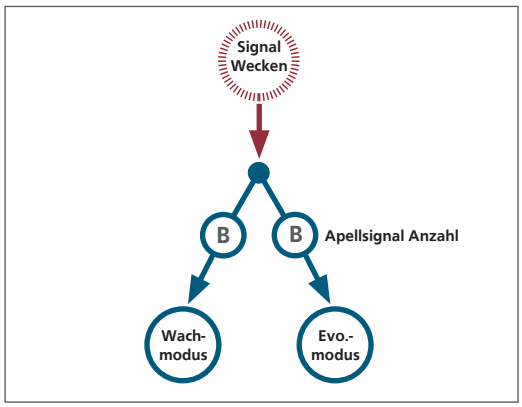


Abbildung 50:
Appellmodus 2 - Entscheidung Evolutions- oder Wachmodus

Werden von allen Einheiten ausreichend Appellsignale empfangen, gehen die gesamten Einheiten in den Wachmodus über. Existieren Einheiten, die von ihren Nachbarn nicht ausreichend Appellsignale empfangen haben, bedeutet dies, dass das System nicht voll betriebsfähig ist. In diesem Fall wird der evolutionäre Entwicklungsprozess gestartet (siehe Abbildung 51).

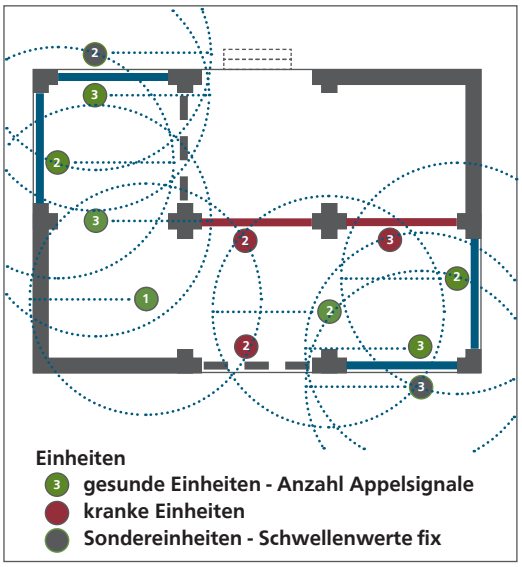


Abbildung 51:
Appellmodus - Apellsignale

Evolutionsmodus

Der Start des Evolutionsmodus beginnt mit der Übermittlung eines Startsignales durch Einheiten, in deren Nachbarschaft nicht aureichend *gesunde* Einheiten existieren (siehe Abbildung 55).
Das Startsignal hat zur Folge, dass sich *kranke* Einheiten auf den Empfang neuer Schwellenwerte einstellen, sie werden zu Empfängern im evolutionären Prozess. Die *gesunden* Einheiten stellen sich auf die Übermittlung ihrer Schwellenwerte ein, sie werden zu Wertespendern (siehe Abbildung 52).

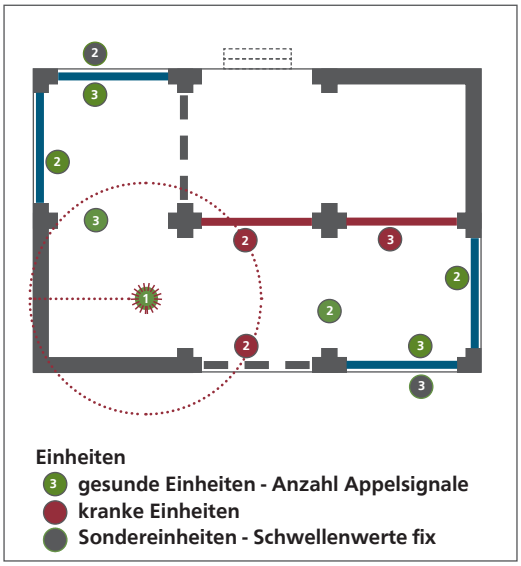


Abbildung 52:
Evolutionsmodus - Initialisierung

Nach einer kurzen Pause, die die Bereitschaft aller Einheiten gewährleisten soll, beginnt der Schwellenwertaustausch. Die Spendereinheiten mutieren zunächst ihre Schwellenwerte geringfügig durch Addierung von Zufallswerten. Dies dient der Aufrechterhaltung der Schwellenwertvarianz. Die mutierten Schwellenwerte werden nun an die Empfänger übertragen. Diese Übertragung ohne Kontrolle erlaubt den Empfang falscher Werte, was ebenso eine erwünschte Mutation darstellt (siehe Abbildung 53).

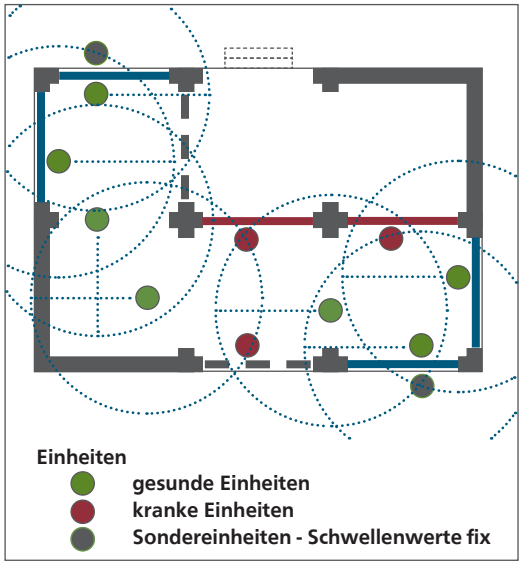


Abbildung 53:
Evolutionsmodus - Schwellenwertübertragung

Nach Abschluss des Schwellenwertaustausches werde aus den Empfängern Einheiten einer neuen Generation mit neuer gesetzter maximaler Lebenszeit (siehe Abbilungen 54, 55).

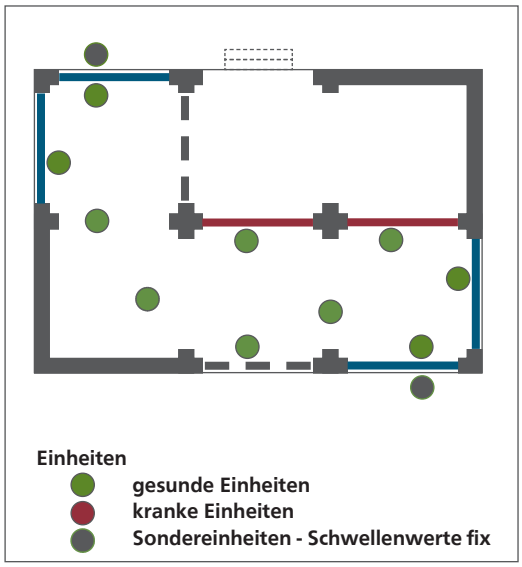


Abbildung 54:
Evolutionsmodus - Neustart mit gesunden Einheiten

Die Einheiten generieren anschließend zur Desynchronisation einen Zufallswert. Dieser ergibt die Länge der Pause vor dem Wechsel der Einheiten in den Wachmodus. Der Start in den Wachmodus erfolgt auf diese Weise mit variierten Pausen, was zu einer Dysnchronisation des Systems führt.

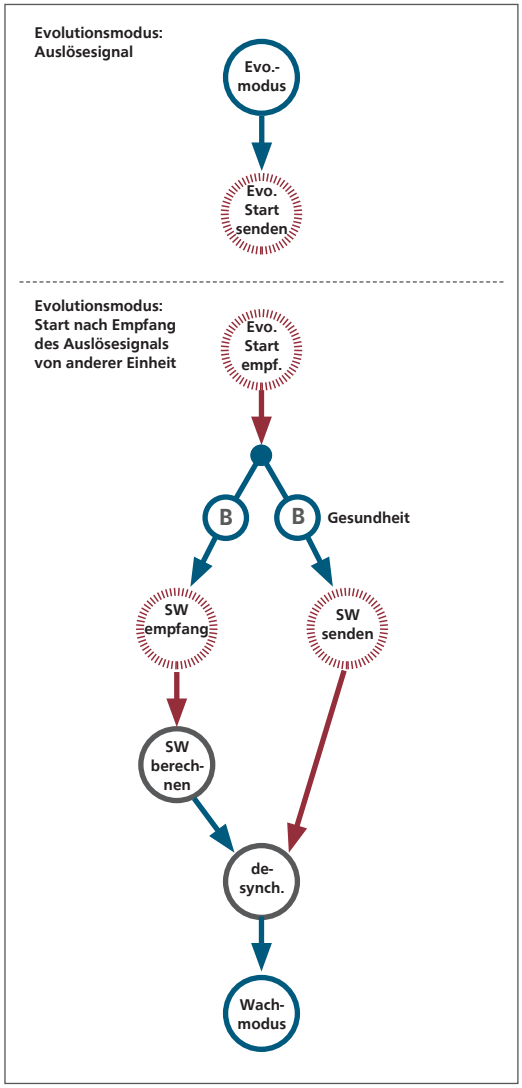


Abbildung 55:
Evolutionsmodus - Auslösung - Ablauf

Wachmodus

Der Wachmodus entspricht dem regulären Tagesbetriebsmodus der ADRRM-Systeme. In diesem Modus liegen die Schwellenwerte der Fuzzy-Neuronen dichter zusammen, was zu hoher Systemaktivität führt. Diese dient der prompten Erfüllung der Nutzeranforderungen. Die Verhaltensweisen einer Einheit sind von den gleichen Faktoren abhängig, die auch zur Modusentscheidung herangezogen werden (siehe

Abbildung 56). Wie bereits erwähnt, unterscheiden sich der Wach- und der Ruhemodus einzig durch ihre Schwellenwerte.

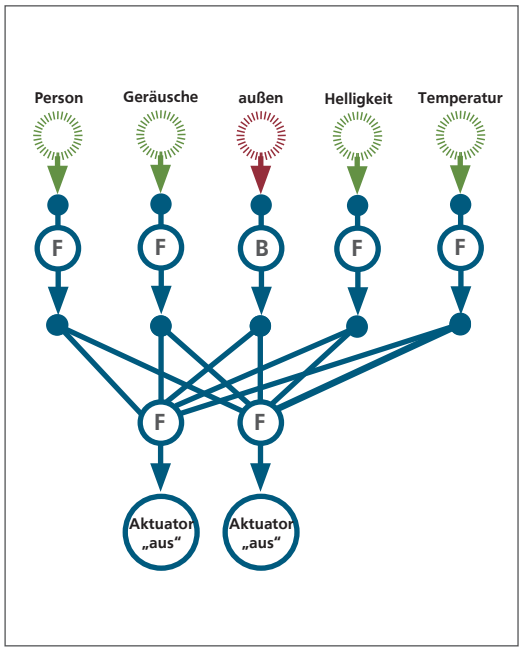


Abbildung 56:
Wach- /Ruhemodus Verhalten

Wechsel vom Wach- in den Ruhemodus

Den Wechsel vom Wach- in den Ruhemodus führen die Einheiten autonom durch. Es kann durchaus sinnvoll sein, dass einige Bereich der gesteuerten Umgebung früher in den Ruhemodus wechseln als andere. Im Gegensatz zum Wechsel vom Ruhe- in den Wachmodus sind bei einem Wechsel vom Wach- in den Ruhemodus keine weiteren globalen Aktivitäten vorgesehen, die eine Synchronisierung dieses Vorgangs erfordern würden.
Auch der Wechsel vom Wach- in den Ruhemodus erfolgt ereignisbasiert. Fällt das Wach-/Ruhepotential aufgrund geringer Umgebungsaktivitäten unter die Wachschwelle, wechselt die Einheit in den Ruhemodus.

Gefahrenmodi

Grundlegend werden in dieser Arbeit drei verschiedene Gefahrenebenen behandelt:

Die Gefahr für Menschenleben, Gefahr für die Umgebungsstruktur und die Gefahren für das Steuerungssystem.

Die Verhaltensweisen zur Gefahrenabwehr sind vorgegeben und unterliegen nicht der evolutionären Entwicklung. Die Maßnahmen sind kooperativ angelegt, die Szenarien wurden vorberechnet oder in Simulationen vor der Montage evolutionär ermittelt. Gefahrensignale werden, abhängig vom Gefahrentypus, von einer Einheit zur anderen weitergeleitet und wirken global. Eine Einheit, deren Sensoren eine dieser Gefahren wahrnehmen, übermittelt ein gefahrenspezifisches Signal. Weitere Einheiten, die ein Gefahrensignal empfangen, arbeiten autonom ein vorgegebenes Abwehrverhalten ab (siehe Abbildung 57). Das System bleibt damit auch nach Ausfall einiger Einheiten handlungsfähig.

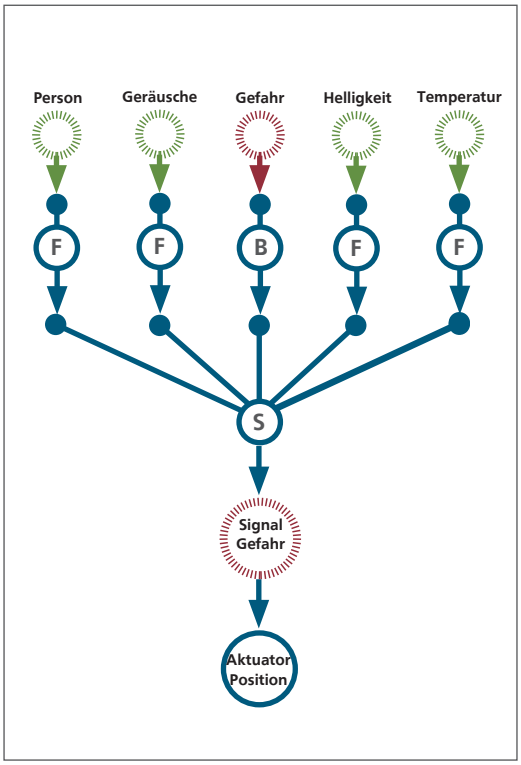


Abbildung 57:
Schema Entscheidung Gefahrenmodus

Die Abwehr von Gefahren für Menschenleben hat oberste Priorität. Treten Gefahren dieser Kategorie

auf, wird der Verlust von Umgebungsstrukturen und der Verlust des Steuerungssystems akzeptiert. Die Abwehrmaßnahmen fallen dementsprechend radikal aus. Zutrittskontrollsysteme werden vom Steuerungssystem übersteuert oder abgeschaltet. Zu den relevanten Gefahren dieser Kategorie gehören unter anderem Feuer oder Erdbeben.

Sind keine Menschen innerhalb der zu steuernden Umgebung anwesend, gilt die Priorität des Systems dem Schutz der Umgebungsstruktur. Einige Gefahren können nun effektiver bekämpft werden, es ist beispielsweise möglich, brennende Bereiche abzuschotten oder die Lüftung abzustellen. Die Sicherheit des Steuerungssystems an sich ist in diesem Fall nachrangig.

Falls keine Gefahren für Menschen und die Umgebungsstruktur bestehen, unternimmt das System auch Selbstschutzmaßnahmen. Die Gefahren bestehen in diesem Fall vor allem in Angriffen auf das Steuerungssystem selbst (siehe den Abschnitt: Abwehr externer Angriffe).

Stellt eine Einheit eine Gefahr fest, übermittelt sie ein gefahrenspezifisches Signal (siehe Abbildung 58).

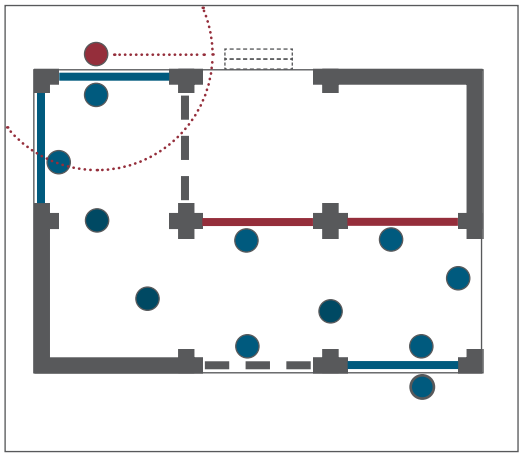


Abbildung 58:
Gefahrenmodus - Teil 1 Initialisierung

Gefahrensignale werden von Einheit zu Einheit übertragen. Die Anzahl der Weiterleitungsschritte ist hierbei entweder sehr hoch oder unbegrenzt eingestellt. Dies hat eine globale Ausbreitung der Gefahrensignale zu Folge (siehe Abbildung 59).

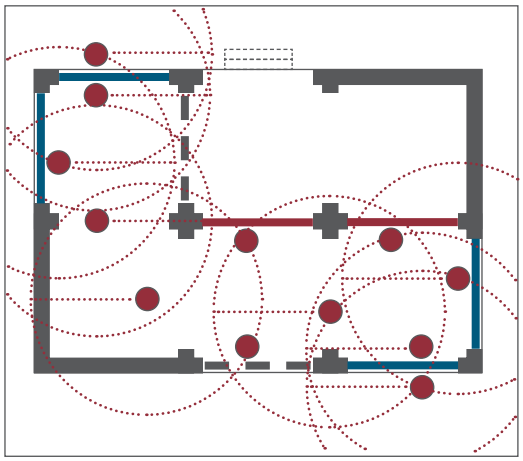


Abbildung 59:
Gefahrenmodus - Teil 2 Ausbreitung

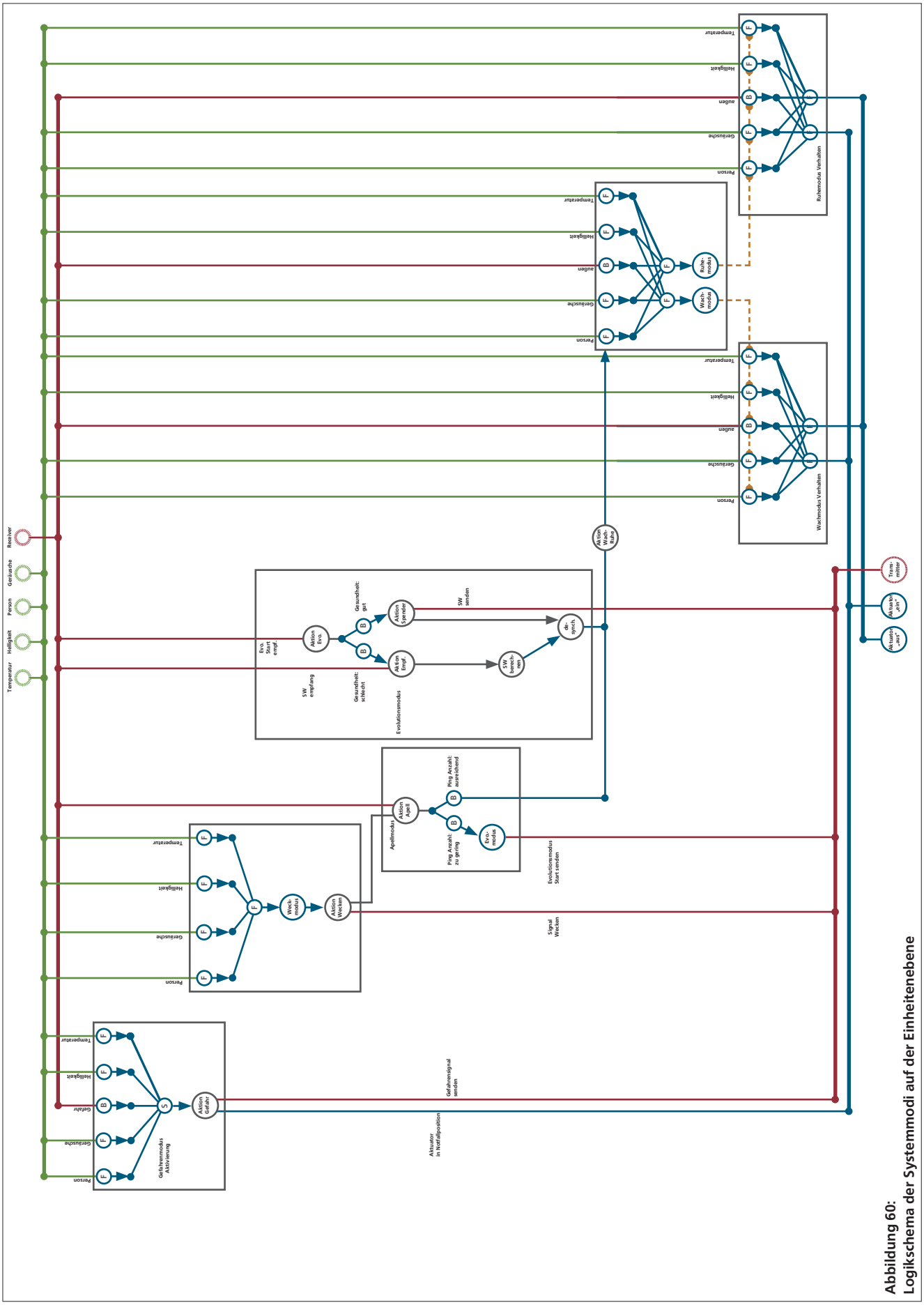


Abbildung 60:
Logikschema der Systemmodi auf der Einheitenebene

3.3.5 Zusammenfassung

Die Strategien und Prinzipien der Metamodelle lassen sich in ein hypothetisches technisches System übertragen.

Das System muss dazu ebenso wie die natürlichen Vorbildsysteme aus einem Kollektiv bestehen. Dieses Kollektiv besteht aus künstlichen autonomen Einheiten, so genannten Agenten. Diese Agenten können über evolutionäre Prozesse selbstanpassend ausgelegt werden. Anpassung über Lernverhalten wird aufgrund der zu hohen Ressourcenanforderungen und der Notwendigkeit vorgegebener Ziele verworfen. Das System lässt sich somit auch in die Kategorie der *Multi-Agenten-Systeme* zuordnen.

Künstliche Kollektive müssen normalerweise über adressierbare Einheiten verfügen, um die Signalausbreitung unter Kontrolle zu halten. Um die Nutzerprivatsphäre zu schützen und das System vor Systemscans zu bewahren, soll auf eine Adressierung der Einheiten verzichtet werden. Dies ist aufgrund des besonderen Einsatzbereiches der Gebäudeautomation bzw. der Automation *urbaner Räume* möglich, da hier auf eine physikalische Signalausbreitungsbegrenzung zurückgegriffen werden kann. Eine physikalische Signalausbreitungbegrenzung kann in einer räumlichen Auflösung umgesetzt werden, die für die beabsichtigten Einsatzbereiche hinreichend ist. Die kleinste umsetzbare Auflösung liegt bei einem 2,4Ghz-System bei einem Radius von ca. einem Meter (siehe Abbildung 61).

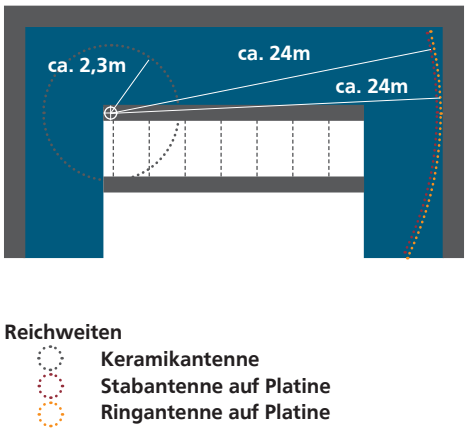


Abbildung 61:
Reichweitentest bei maximaler Sendeleistung Luftraum Architekturgebäude

Dies entspricht in etwa dem Radius eines Arbeitsplatzes. Die größte Reichweite, die in dem Test erreicht werden konnte, betrug 24m. Es können aber auch Reichweiten von ca. 200m realisiert werden, die dann auch für den Einsatz in urbanen Dimensionen ausreichend wären.[1]

Es ist somit auch in einem dezentralen System ohne adressierbare Einheiten möglich, individuellen räumlich begrenzten Einfluss zu nehmen, indem man über eine mobile Einheit zu einem Mitglied des Kollektivs wird. Die Mobileinheiten kommen dabei ebenfalls ohne eine Adressierung aus und sind damit ebenfalls vollständig anonym.

Da sämtliche Einheiten anonym ausgelegt und zentrale Schnittstellen weder nötig noch vorhanden sind, schützt das System die Nutzerdaten auf der elementarsten Ebene. Es sind schlicht keine Nutzerdaten vorhanden. Das System ist zudem unzugänglich und selbst ein Abhörangriff von außen könnte keine zuordbare Daten erfassen. Das System ist damit extrem nutzerdatensicher.

Ein Steuerungskollektiv aus autonomen aber kooperativ zusammenarbeitenden Agenten ist in der Lage, viele Aufgaben parallel zu bearbeiten. Es ist zudem in der Lage, auch umfangreiche Aufgaben zu erfüllen, wenn es gelingt, diese in Form von Regeln auf das Kollektiv zu verteilen.

Die Verteilung globaler Aufgaben auf ein Kollektiv kann, wie oben erwähnt, über Regeln erfolgen. Die Regeln bestimmen die Verhaltensweisen der Einheiten, die in ihrer Gesamtheit wiederum das globale Systemverhalten ausprägen. Die Regeln können innerhalb der Einheiten in Regelkreisen angelegt werden. Diese Regelkreise müssen teilweise in der Lage sein, flexibel zu reagieren, um ein hyperaktives Systemverhalten zu vermeiden. Da die Verhaltensweisen der Einheiten zudem von multiplen Faktoren abhängig sind, müssen die Regelkreise in der Lage sein, multiple Signale zu verarbeiten.

Die Verarbeitung multipler Signale zu einem Ausgangssignal innerhalb eines Elements legt den Ver-

1 Vgl. Sparkfun Electronics: How Far Does It Go? (http://www.sparkfun.com/tutorials/48), 2. Juni 2011.

gleich mit einem Neuron nahe. Diese Regelkreise können daher als künstliche Neuronen bezeichnet werden. Die multiplen Einflüsse wirken dabei sich hebend oder senkend auf das Neuronenpotential aus.

Neuronen, die der Entscheidungsfindung dienen werden, sollten flexibel angelegt werden. Dies kann über die Einführung zweier Schwellenwerte pro Neuron geschehen. Diese Schwellenwerte werden mit einigem Abstand zueinander angelegt und wirken entweder nur bei Unter- oder Überschreitung der Schwelle. Dies ergibt einen Bereich, in dem die vorhergehende Entscheidung erhalten bleibt; eine direkte Zuordnung des Potentialwerts zu einer bestimmten Aktion existiert in diesem Bereich nicht. Es handelt sich somit um weiche Schwellen und damit um einen Ableger der *Fuzzy-Logik*.

Das Zusammenwirken einer Vielzahl von Einheiten, deren Verhaltensweisen wiederum von einer Vielzahl von Faktoren abhängig ist, wird ab einer bestimmten Systemgröße deterministisch chaotische Zustände aufzeigen. Das bedeutet, das globale Systemverhalten wird nicht vorhersehbar erscheinen, obwohl die Programmierung der Einheiten klar deterministisch ist. Das zu beobachtende *Chaos* tritt einzig aufgrund der Nichtwiederholbarkeit der Abhängigkeiten bzw. der äußeren und inneren Umstände auf. Für einen Beobachter wird *deterministisch chaotisches Systemverhalten* zumindest teilweise *(schwach-) emergent* erscheinen. Das heißt, er wird globale Verhaltensweisen beobachten, die er selbst bei genauer Kenntnis des Aufbaus und der Programmierung der Einheiten nicht vorhersehen kann. Eine der grundsätzlichen Schwierigkeiten wird darin bestehen, Regeln zu finden, die ein sinnvolles emergentes Verhalten generieren.

Um das Verhalten der Einheiten und damit auch das globale Verhalten zu kontrollieren, können Betriebsmodi eingerichtet werden. Diese Modi strukturieren das Verhalten der Einheiten abhängig von den Umgebungsfaktoren und beschränken die Handlungsfreiheit der Einheiten auf die jeweils situationsgerechten Reaktionen. Anhand der Modi kann das System auch Selbstanalysen und Anpassungsprozesse vornehmen. Durch die über den Tag wechselnden Betriebsmodi wird das Verhalten strukturiert und die Emergenz des globalen Systemverhaltens begrenzt.
Es ist ersichtlich, dass zumindest theoretisch die Möglichkeit besteht, ein technisches System zu konstruieren, das den Anforderdungen der These gerecht wird.

3.4 Die Versuchseinheiten

3.4.1 Aufbau der Versuchseinheiten

Zur Überprüfung der These wurde ein künstliches Steuerungskollektiv entwickelt (siehe Kapitel 3.6.1 Versuchsaufbau). Dieses Kollektiv setzt sich aus den vom Autor selbst entwickelten und programmierten Einheiten zusammen. Mit diesem Steuerungskollektiv bzw. mit diesem ADRRM-Prototypensystem sollte der Nachweis der grundlegenden Funktionstüchtigkeit des theoretischen technischen System erbracht werden.

Als pheromonähnliches Kommunikationsmittel wurde Funk gewählt. Akustische oder olfaktorische Kommunikationsmittel erschienen technisch unpraktikabel.
Funkwellen bzw. elektromagnetische Wellen lassen sich in ihrer Ausbreitung durch eine Regulierung der Sendeleistung begrenzen. Elektromagnetische Wellen erfüllen damit die Grundvoraussetzung zur Anwendung der Kommunikationsprinzipien der staatenbildenden Insekten. Die Einheiten mussten somit über einen Funksender und -empfänger verfügen. Diese beiden Funktionen wurden von einem so genannten Transceiver übernommen. Die Wahl fiel hierbei auf einen Nordic nRF2401A Transceiver der Firma Sparkfun.[1]

Zudem mussten die Einheiten über einen Logikbaustein verfügen, der die Datenverarbeitung übernahm. Hierfür wurden Prototypingboards aus dem Arduino- Entwicklungssystem benutzt.[2]

Jede Einheit erhielt als Standardausrüstung einen Temperatursensor und einen Helligkeitssensor. Je nach Einsatzzweck wurden Infrarotsensoren oder Windsensoren ergänzt.

Die Einheiten waren in der Lage analoge Aktuatoren, RGB-Leds oder Servos, über PWM[3]-Ausgänge zu steuern. Des Weiteren besaßen die Einheiten einen digitalen Schaltausgang, mit dem Relais oder Leds geschaltet werden konnten.
Die Einheiten wurden mit 3,3 Volt betrieben und benötigten bei voller Auslastung maximal 0,23 Watt/h. Als Spannungsversorgung wurden 1000mW Lipo Akkus, Blei-Gel-Akkus und ein Schaltnetzteil eingesetzt (siehe Kapitel 3.5.1 Versuchsaufbau).

Entwickungsgeschichte der Versuchseinheiten

Zunächst wurden die Bauteile auf einem Steckbrett - Breadboard - zu einer Schaltung zusammengefügt. (siehe Abbildung 62).

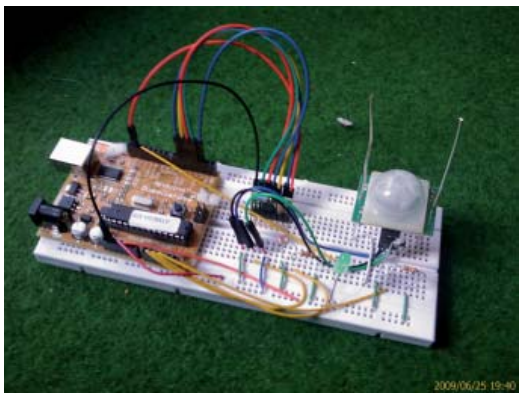


Abbildung 62:
Breadboard Prototyp
Arduino Duemillanove, Nordic Transceiver, Sensoren

Die funktionsfähige Schaltung wurde dann auf ein Platinenlayout übertragen (siehe Abbildungen 63, 64).

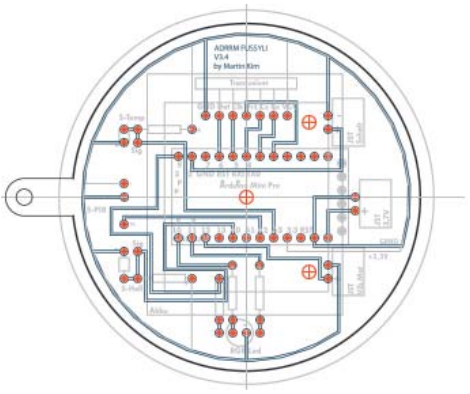


Abbildung 63:
Platinenlayout Oberseite

1 Webseite der Firma Sparkfun, (http://www.sparkfun.com/), 23. Mai 2011.

2 Webseite der Arduino Community, (http://www.arduino.cc/), 23. Mai 2011.

3 PWM: Pulsweitenmodulation.

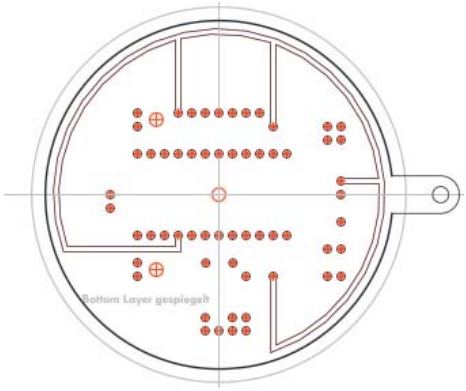


Abbildung 64:
Platinenlayout Unterseite

Von diesem Layout wurden zwei Prototypen gefräst, einer an der TU-Darmstadt (siehe Abbildungen 65, 66)

Abbildung 65:
Fräse an der Technischen Universitit Darmstadt

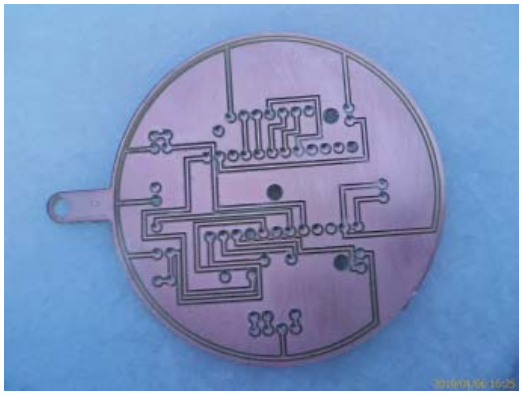


Abbildung 66:
gefräster Prototyp von der TU-Darmstadt, Oberseite

und einer an der RWTH-Aachen (siehe Abbildungen 67, 68).

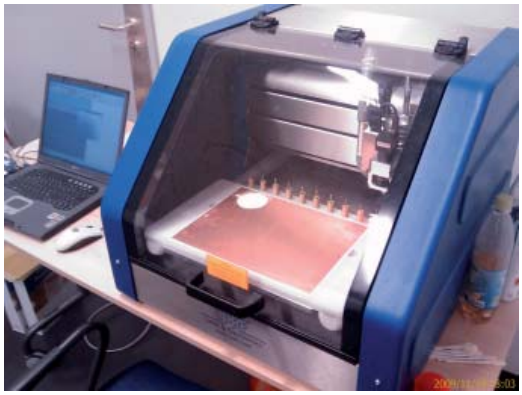


Abbildung 67:
Fräse an der RWTH Aachen
Arduino Pro Mini, Nordic Transceiver, Sensoren

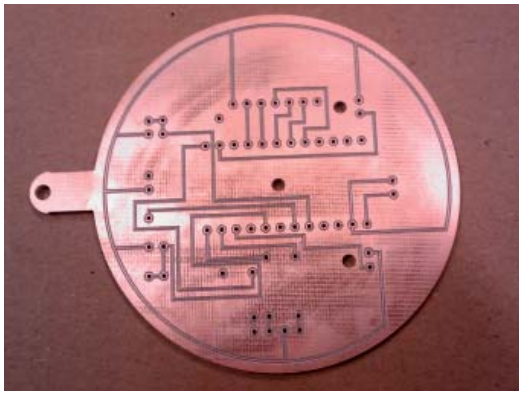


Abbildung 68:
gefräster Prototyp von der RWTH Aachen, Oberseite

Auf Grundlage dieser Prototypen wurde eine Kleinserie von zwanzig Stück in Auftrag gegeben (siehe Abbildung 69).

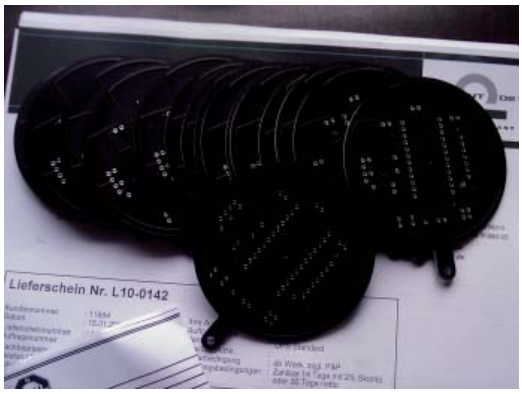


Abbildung 69:
Platinen der Kleinserie

Aufbau der Serieneinheiten

Die Steuerungseinheiten der Kleinserie wurden auf der Platinenoberseite mit dem Nordic Transceiver bestückt, der hier in der Variante mit Stabantenne abgebildet ist - die rot-gelbe Platine. Quer dazu liegt der Ardunio Pro Mini - die blaue Platine. Unter der Montageöse befinden sich die Sensoren. Die Steuerungseinheiten wurden mit einem Helligkeitssensor, links im Bild, und einem Temperatursensor, mittig unter der Befestigungsöse, ausgestattet. Links im Bild sind die PWM-Schnittstellen sichtbar, an denen die Servos oder eine RGB-Led angeschlossen werden konnten. Unten im Bild sind die Pins der Programmierschnittstelle zu sehen (siehe Abbildung 70).

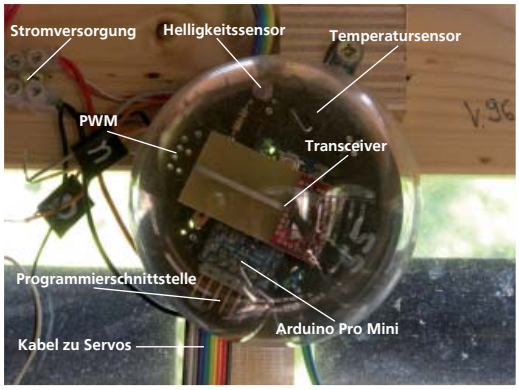


Abbildung 70:
Steuerungseinheit Oberseite

Auf der Unterseite befinden sich Buchsen für zwei Digitalausgänge und mittig eine Buchse für die Spannungsversorgung (siehe Abbildung 71).

Abbildung 71:
Anschlüsse Unterseite

Spezialisierte Einheiten

Ergänzend zu den Steuerungseinheiten wurden einige spezialisierte Einheiten konstruiert.

Personendetektionseinheiten

Drei Einheiten wurden mit Passiv-Infrarotsensoren zur Bewegungsdetektion ausgerüstet (siehe Abbildung 72).

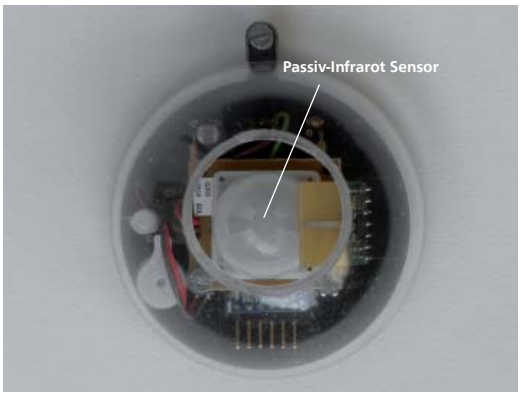


Abbildung 72:
Personendetektionseinheit Oberseite

Eine dieser Einheiten wurde zur Erfassung von Bewegungen im Außenbereich genutzt (siehe Abbildung 73).

Abbildung 73:
Personendetektionseinheit außen

Bewegungen im Außenbereich lösten den Eindringlingsalarm aus.

Eine Personendetektionseinheit wurden zur Personenerfassung im Innenraum eingesetzt (siehe Abbildung 74). Die Anwesenheit von Personen im Innenbereich hielt das System im Wachmodus. Die dritte Einheit wurde ebenfalls im Außenbreich eingesetzt, fiel aber nach kurzer Zeit der Witterung zum Opfer.

Abbildung 74:
Personendetektionseinheit außen

Abbildung 76:
Mobileinheit

Funkbrückeneinheiten

Zwei Einheiten wurden als Funkbrücken in Bereichen eingesetzt, in denen aufgrund zu weniger Steuerungseinheiten Lücken in der Funkabdeckung auftraten. Diese Einheiten wurden mit RGB-Leds anstatt der Servoanschlüsse ausgestattet die den aktuellen Funkverkehr visualisierten (siehe Abbildung 75).

Abbildung 75:
Funkbrücke

Akustische Feedbackeinheit

Eine weitere Einheit, die der Kommunikation zwischen Nutzer und System diente, war die akustische Feedback-Einheit. Diese Einheit besaß einen Lautsprecher, der den Funkverkehr akustisch abbildete (siehe Abbildung 77).

Abbildung 77:
Feedback-Einheit mit Lautsprecher

Mobileinheiten

Eine Einheit wurde zur Mobileinheit ausgebaut. Mobileinheiten konnten vom Nutzer mitgeführt werden. Diese Mobileinheiten ermöglichten es dem Nutzer ein Teil des Systems zu werden. Als Mitglied des Kollektivs konnte der Nutzer mit der Mobileinheit Werte in das System einspeisen. Die Einspeisung eigener Werte war gleichbedeutend mit einer individuellen Einflussnahme auf die Umgebung. Der Nutzer konnte somit seine Wünsche an das System übermitteln. Die Mobileinheiten besaßen Taster, mit denen der Wunsch nach mehr oder weniger Helligkeit signalisiert werden konnte (siehe Abbildung 76).

Die Signale der Einheiten ergaben ein charakteristisches Klangbild.[4] Änderungen in diesem Klangbild wiesen auf besondere Vorkommnisse, beispielsweise Gefahren oder Ausfälle von Einheiten, hin. Anhand von Akustikeinheiten oder anderen *Feedbackeinheiten* ist es beispielsweise möglich, umfangreiche Umgebungen effizient auf Unregelmäßigkeiten zu untersuchen.

4 Vom Autor: Klangbild. (http://vimeo.com/22550950) 6.Juni 2011.

Äußere Sensoreinheiten

Zwei Einheiten wurden als Sensoreinheiten im Außenbereich eingesetzt. Diese Einheiten wurden mit Ringantennen ausgerüstet, die eine etwas größere Reichweite als die Stabantennen hatten. Da diese Einheiten keine Servos steuern mussten, konnten sie mit RGB-Leds zur Statusanzeige bestückt werden (siehe Abbildung 78).

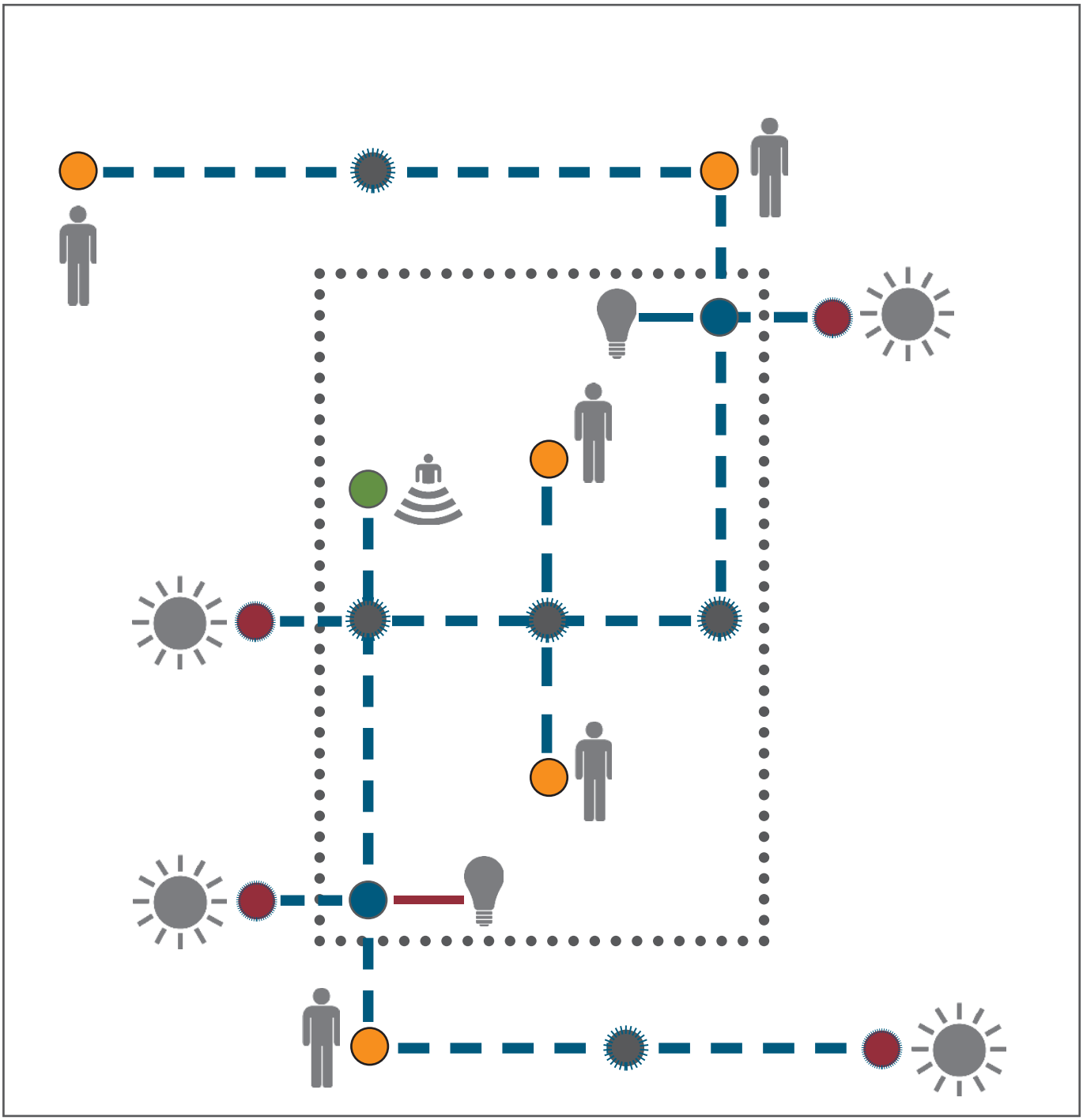
Abbildung 78:
Sensoreinheit außen

Die nördliche Sensoreinheit wurde zusätzlich an einen Windsensor angeschlossen (siehe Abbildung 79).

Abbildung 79:
Windsensor außen

Stieg die Windgeschwindigkeit über ein eingestelltes Maß an, wurde ein Gefahrenalarm ausgelöst. Dieser Windalarm führte zur Sicherung des Versuchsgebäudes über die Schließung aller Klappen.

Beispielhaftes ADRRM-Vernetzungsschema



Einheiten
- Steuereinheiten
- Funkbrücken
- Personendetektion
- Wind- und Außenwerte
- Mobileinheit

Abbildung 80:
ADRRM-Vernsetzungsschema

3.4.2 Programmierung der Versuchseinheiten

An dieser Stelle wird ausschließlich die Grobstruktur der Programmierung erläutert. Das vollständige Programm mit Erläuterungen findet sich im Anhang 7.2.

Programmablauf (siehe Abbildung 81)

Zu Beginn des Programms wird der Empfang von Funksignalen abgefragt.
Werden keine Funksignale empfangen, wird im Falle eines Programmneustarts die *Lebenszeitbegrenzung* auf die maximale Lebenszeit gesetzt. Erfolgte kein Programmneustart wird die verbleibende Lebenszeit berechnet.

In einem nächsten Schritt wird die Hauptroutine *Haupt 1* aktiviert.
Ist die Einheit aufgrund hoher Umgebungsaktivität von selbst erwacht, geht sie in den *Apellmodus*. Zu Beginn dieses Modus wird ein Wecksignal gesendet. Dieses Wecksignal löst einen systemweiten Zählapell aus. Fällt dieser Zählapell aufgrund zu weniger aktiver bzw. *gesunder* Einheiten negativ aus, wird der *Evolutionsmodus* aktiviert. Der *Evolutionsmodus* bewirkt einen Austausch von Schwellenwerten zwischen *gesunden* und *kranken* Einheiten.

Wird ein Funksignal empfangen, werden zunächst die empfangenen Signale abgespeichert. Ist dies erfolgt, wird auch hier die verbleibende *Lebenszeit* berechnet.
Es folgt die Aktivierung der Hauptroutine *Haupt 2.*
Wird ein Gefahrensignal empfangen, geht die Einheit direkt in den entsprechenden *Gefahrenmodus*. Die *Gefahrenmodi* aktivieren die entsprechenden voreingestellten Aktionen zur Gefahrenabwehr. Sobald die Gefahr übertstanden ist, geht die Einheit in den *Wach-Ruhemodus*.
Handelt es sich bei dem empfangenen Signal nicht um ein Gefahrensignal, wird zunächst geprüft, ob es sich um ein *Wecksignal* handelt. Ist dies der Fall, geht die Einheit in den *Apellmodus*.
Sollte das empfangene Signal weder Gefahrensignal noch Wecksignal sein, wird geprüft, ob es sich um ein *Pingsignal* zur Ermittlung der minimal notwendigen Sendeleistung handelt. Einem *Anfrageping* wird mit einem *Antwortping* beantwortet.
Handelt es sich bei dem empfangenen Signal nicht um ein Pingsignal, wird geprüft, ob es sich um ein *Informationssignal* handelt. Sollte auch dies nicht zutreffen, ist das Signal fehlerhaft oder aus fremder Quelle. Signale dieser Art werden ignoriert, und die Einheit geht in diesem Fall in den *Wach-Ruhemodus*.
Handelt es sich hingegen um ein *Informationssignal* einer anderen Einheit, werden die Werte gespeichert und an den *Wach- Ruhemodus* weitergeleitet.

Die Einheit geht nun in den *Wach-Ruhemodus*. In dieser Routine wird entschieden, ob die Einheit in den Wach- oder den Ruhezustand übergeht. Umdiese Entscheidung zu treffen, werden zunächst die Sensoren abgefragt. Die Sensorenwerte werden dann mit dem *Gesundheitspotentia*l verrechnet, das über den *Gesundheitszustand* der Einheit entscheidet. Fällt dieses Potential unter eine kritische Schwelle, gilt die Einheit als krank und die Einheit wird im nächsten Evolutionsmodus neue Schwellenwert empfangen. Ist die Einheit *gesund*, wird sie Schwellenwerte spenden.
Im nächsten Schritt werden die Sensorwerte mit dem *Wach-Ruhepotential* verrechnet. Fällt dieses Potential unter die *Wachschwelle*, begibt sich die Einheit in den *Ruhemodus*. Verbleibt das Potential über der *Wachschwelle,* wechselt die Einheit in den *Wachmodus*. Den *Wach-* und den *Ruhemodus* unterscheiden einzig unterschiedliche Schwellenwerte zur Aktivierung der Aktuatoren. Im *Wachmodus* liegen die Schwellenwerte nahe beieinander, woraus ein dynamisches Verhalten resultiert. Der *Ruhemodus* besitzt weit gespreizte Schwellenwerte, die träge Reaktionen zur Folge haben.
Im folgenden Schritt wird über einen Zufallswert ermittelt, ob eine *Reichweitenanpassung* erfolgen soll. Ist dies der Fall, wird ein *Pingsignal* mit der niedrigsten Sendeleistung gesendet und auf Anwort gewartet. Empfängt die Einheit *Antwortpings* von mindestens zwei anderen Einheiten, bleibt sie auf der niedrigsten Sendeleistung. Wird keine ausreichende Anzahl von *Antwortpings* empfangen, wird der Vorgang mit erhöhter Sendeleistung wiederholt.
In einem letzten Schritt wird geprüft, ob die *Lebenszeit* abgelaufen ist. Ist dies der Fall, wird die Einheit zwangsweise auf *krank* gesetzt und erhält als *Empfänger* neue Schwellenwerte während des nächsten *Evolutionsmodus*.

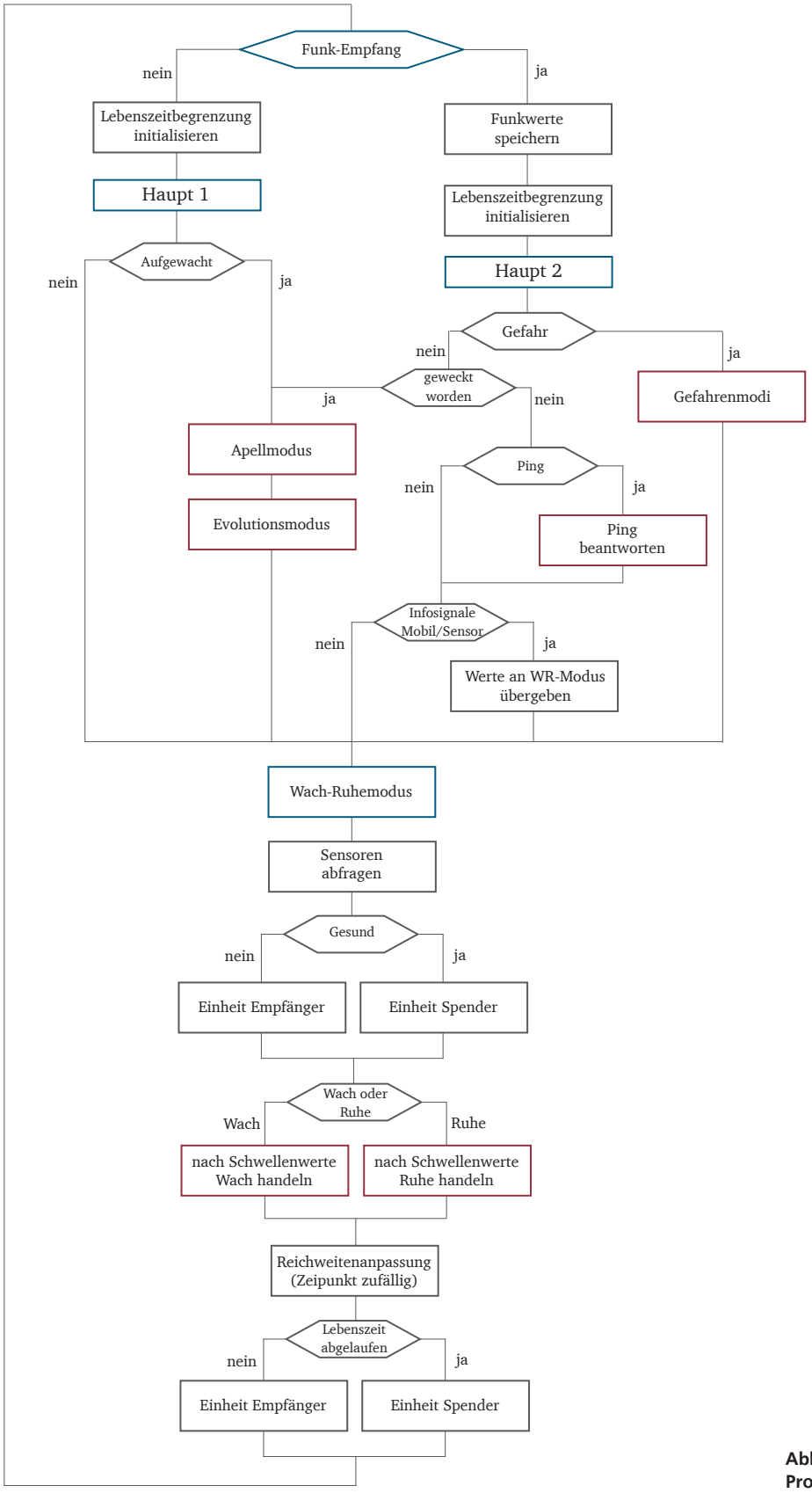


Abbildung 81:
Programmschema ADRRM

3.5 Versuchsaufbau - Versuchsablauf

3.5.1 Versuchsaufbau

Als Versuchsträger diente ein vorhandener Rohbau aus massivem Klinkermauerwerk auf dem Experimentierfeld des Fachbereichs Architektur der TU-Darmstadt. Das Experimentierfeld befindet sich am östlichen Ende des Lichtwiesenwegs (siehe Abbildungen 81, 82).

Abbildung 81:
Lageplan, Versuchgebäude auf dem Experimentierfeld des FB Architektur der TU-Darmstadt - genordet

Abbildung 82:
Südostseite des Mauerwerksversuchsgebäudes Experimentierfeld FB Architektur TU-Darmstadt

Der Mauerwerksrohbau wurde mit Wänden versehen. Diese Wände wurden mit steuerbaren Klappen zur Verschattung und Belüftung ausgestattet (siehe Abbildungen 83, 84).
Die Klappen wurden über Servos bewegt (siehe Abbildung 85). Die Servos wurden pro Wandmodul in zwei getrennten Gruppen angesteuert. Jedes Wandmodul erhielt eine eigene Steuerungseinheit.

Abbildung 83:
Mauerwerksgebäude mit Versuchsfassade

Abbildung 84:
Innenansicht

Abbildung 85:
Servo zur Betätigung einer Verschattungsklappe
Servos zur Betätigung der Lüftungsklappen

Der Mauerwerksbau wurde mit acht Wandmodulen geschlossen, so dass ein L-förmiger Raum entstand (siehe Abbildung 86).

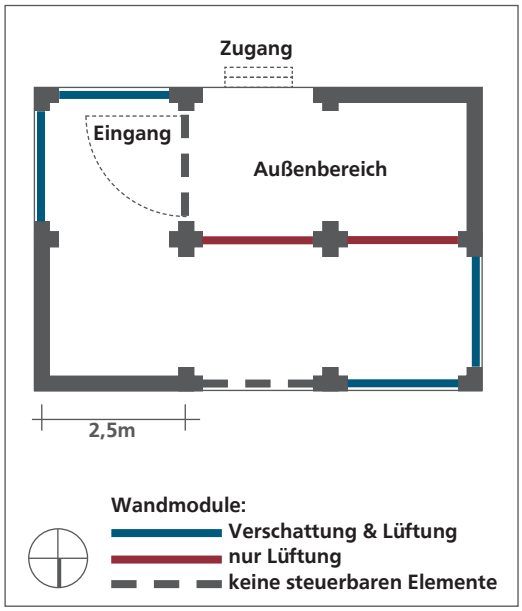


Abbildung 86:
schematischer Grundriss des Versuchsgebäudes

Das Versuchsgebäude wurde mit maximal vierzehn Einheiten geregelt; inklusive der Mobil- und der Feedbackeinheit.
Die sechs mit Klappen ausgestatten Module erhielten jeweils eine individuelle Steuerungseinheit (Abbildung 87).

Abbildung 87:
Steuerungseinheit

Das Modul mit der Eingangstür und das mittlere im Norden gelegene Modul besaßen keine Klappen (siehe Abbildung 88). Es wurden zwei Funkbrücken eingesetzt, um Lücken in der Funkabdeckung abzudecken. Über dem mittleren Modul auf der Nordseite wurde eine Personendetektionseinheit angebracht.

Im Außenraum wurden zwei Sensoreinheiten (siehe Abbildung 89) montiert, von denen die nördliche mit einem Windsensor verbunden war (siehe Abbildung 90).

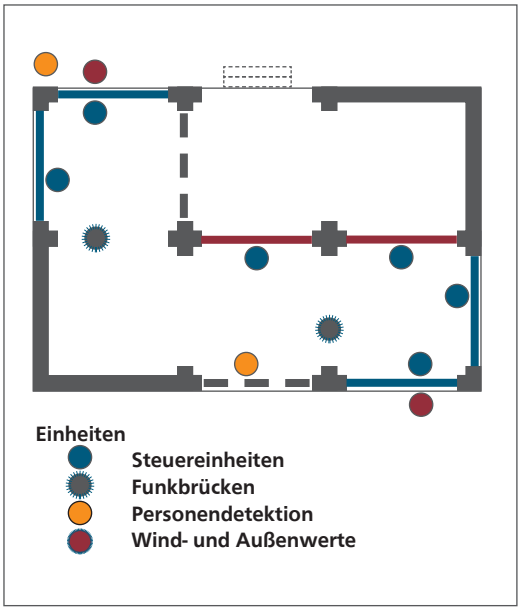


Abbildung 88:
Verteilung der Einheiten

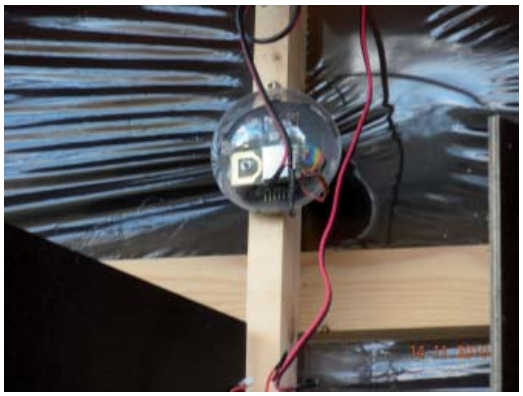


Abbildung 89:
Standardeinheit als äußere Sensoreinheit
hier mit Transceiver mit Ringantenne

Abbildung 90:
Windsensor

An der süd-westlichen Ecke wurde eine weitere Personendetektionseinheit befestigt (siehe Abbildung 91).

Abbildung 91:
äußere Personendetektionseinheit

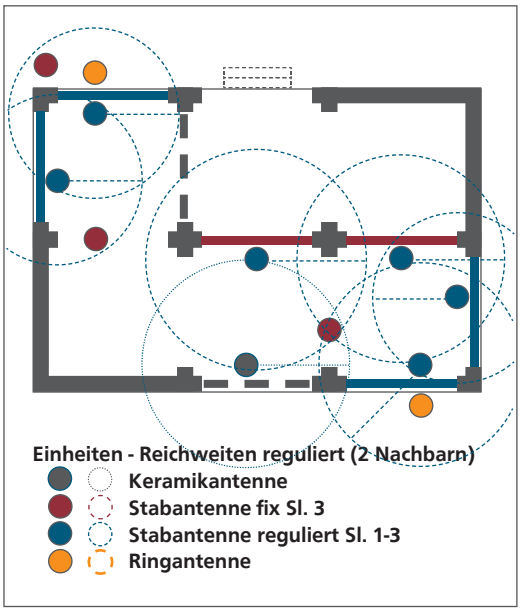


Abbildung 92:
Steuerungseinheiten maximale Reichweiten

Die Reichweiten der einzelnen Einheiten wurden auf sich überlappende aber nicht den gesamten Raum umfassende Bereiche angelegt. Hierbei galt es zusätzlich zu beachten, dass die Signale der Transceiver, die auf einer Frequenz von 2,4Ghz sendeten, Mauerwerk nicht durchdringen konnten. Mauern verursachten dementsprechend Funkschatten, die von anderen Einheiten überbrückt werden mussten. Hohe Luftfeuchtigkeit wirkte sich negativ auf die Sendereichweite aus. Die Luftfeuchtigkeit war zudem großen Schwankungen unterworfen. Somit schwankte auch die Sendereichweite bei gleichbleibender Sendeleistung. Die Sendeleistung musste daher ständig an die jeweilige Luftfeuchtigkeit angepasst werden.

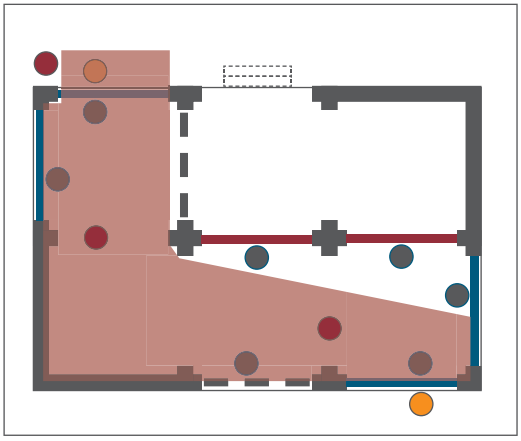


Abbildung 93:
Brückeneinheit-1 maximale Reichweite

Wie aus der Abbildung 92 ersichtlich ist, ergaben sich Lücken in der Funkabdeckung. Es bestand keine Verbindung von dem kurzen Schenkel des L-förmigen Raumes zu dem langen Schenkel. In der Folge besaß nicht jede Einheit die erforderlichen zwei Nachbarn in direkter Reichweite. Daher mussten zusätzliche Funkbrücken eingesetzt werden um diese Lücken zu schließen.
Die Funkbrücken wurden ebenso wie die Steuerungseinheiten mit Stabantennen ausgerüstet, ihr Senderadius wurde aber durch Mauerwerk stark unterbrochen (siehe Abbildungen 93, 94).

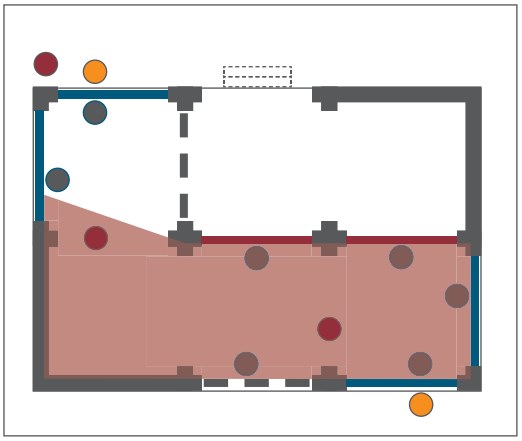


Abbildung 94:
Brückeneinheit-2 maximale Reichweite

Die äußeren Sensoreinheiten waren mit Ringantennen ausgerüstet, die eine etwas größere Reichweite besaßen (siehe Abbildung 95,96).

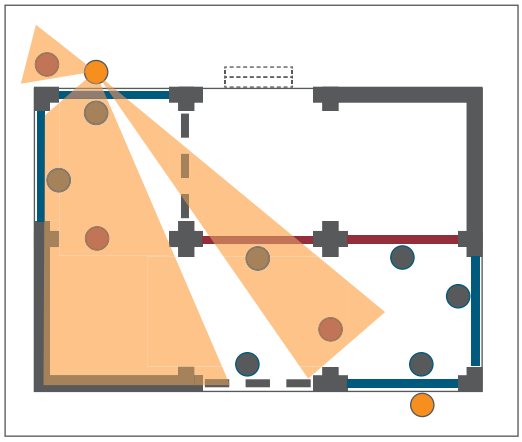


Abbildung 95:
Brückeneinheit-1 maximale Reichweite

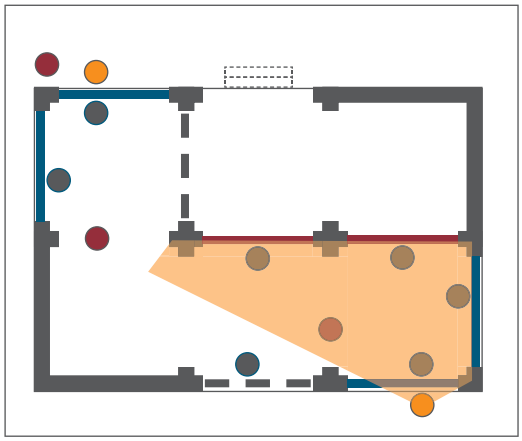


Abbildung 96:
Brückeneinheit-2 maximale Reichweite

In der Kombination ergab sich bei niedriger Luftfeuchtigkeit die folgende Reichweitenverteilung (siehe Abbildung 97).

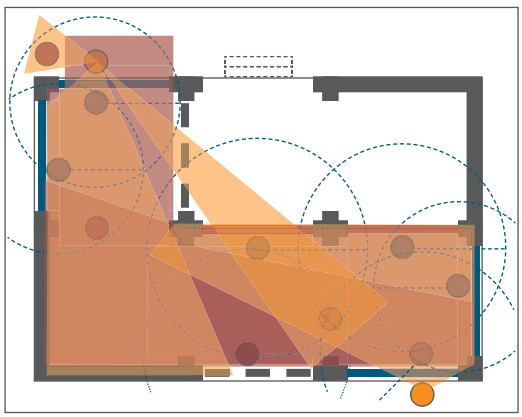


Abbildung 97:
maximale Reichweiten bei niedriger Luftfeuchtigkeit

Die Versuchsdaten wurden an zwei Einheiten des Versuchaufbaus erfasst, an der Steuerungseinheit des Modul 1 und an der Steuerungseinheit des Modul 6 (siehe Abbildung 98).

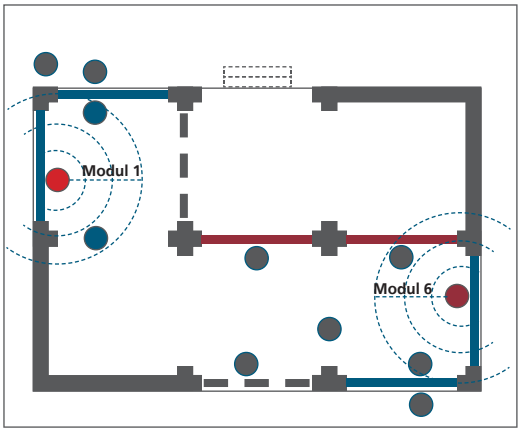


Abbildung 98:
Einheiten die zum Logg herangezogen wurden

Erfasst wurden die gesendeten und empfangenen Funksignale, Sensordaten und verschiedene Potentiale (siehe Kapitel 3.7.3 Logfiles und Interpretation). Zur Datenerfassung diente ein Laptop mit einer Logsoftware zur Aufzeichnung der seriellen Daten der Steuerungseinheiten (siehe Abbildungen 99, 100).

Abbildung 99:
Messung; Notebook ist in der Box mit Insektennetz

Abbildung 100:
Serial Logger

Da keine Gebäude mit vorhandenen steuerbaren Elementen zur Verfügung standen und ein Test in einem im Betrieb befindlichen Gebäude zu problematisch erschien, ergab sich einzig die Option, den vorhandenen Mauerwerksrohbau auf dem Experimentierfeld des FB-Architektur zu nutzen und mit steuerbaren Elementen zu versehen.
Der Versuchsaufbau auf Basis des vorhandenen Mauerwerksrohbaus hatte allerdings einen gravierenden Nachteil. Der Rohbau war nicht gedämmt und verfügte auch nicht über eine Heizungsanlage. Der Einfluss des Steuerungssystems auf das Innenraumklima war daher nahezu inexistent. Gerade in der Datenerfassungsphase, der so genannten *Loggphase*, die im Herbst und Winter stattfand, waren Innen- und Außentemperatur nahezu identisch. Diese Tatsache verhinderte einen Nachweis des Entwicklungspotentials des evolutionären Mechanismus.

3.5.2 Versuchsablauf

Der Versuch diente ausschließlich der Überprüfung der Systemprinzipien, *Proof-of-Concept*, und nicht einem Test der Alltagtauglichkeit. Der Versuchsablauf erfolgte in drei Phasen.

Phase eins - Debugging und Einstellungen

Nach der Montage des ADRRM-Systems in dem Mauerwerksrohbau erfolgte eine mehrmonatige Testphase. Diese Phase umfasste den Zeitraum von August bis November des Jahres 2010. Während dieser Phase wurde die Programmierung und die Hardware auf Fehler untersucht. Es traten diverse Probleme im technischen Bereich auf, unter anderem kam es zu Kurzschlüssen durch *reale Bugs*[1] (Käfer). Es konnten nahezu alle Probleme der Programmierung behoben werden. Allerdings verblieb bei der Übermittlung der evolutionären Schwellenwerte ein Fehler, der erst nach der Loggphase behoben werden konnte.
In dieser Phase wurden auch die Starteinstellungen der Schwellenwerte ermittelt. Diese Werte mussten an die jahreszeitlichen Bedingungen angepasst werden. In den Wintermonaten machte sich die fehlende Isolierung über extrem niedrige Temperaturwerte im Innenraum bemerkbar, denen das System ohne Heizung nicht entgegenwirken konnte.

1 Vgl. MARX, Christy: Grace Hopper. The First Woman to Program the First Computer in the United States. New York 2004,

Nachdem die Programmierung weitgehend korrigiert (*debugged)* worden war, konnte die eigentliche Datenerfassungsphase (*Logphase*) beginnen.

Phase zwei Logphase

Anhand des debuggten Systems konnten nun die Messungen erfolgen, die zum Nachweis der Funktionstüchtigkeit des Systems benötigt wurden.
Die Daten wurden in zwei unterschiedlichen Messszenarien erfasst. Zur Erfassung allgemeiner Systemvorgänge wurden über einen längeren Zeitpunkt Daten einer Einheit erfasst und aufgezeichnet, ein Vorgang, der als *loggen* bezeichnet wird. Um ein räumlich differenziertes Systemverhalten zu erfassen, wurde gleichzeitig an zwei weit auseinanderliegenden Einheiten geloggt. Da es sich auch bei dem Versuchssystem um ein vollständig dezentrales System handelte, war eine Datenerfassung des gesamten Systems nicht möglich.

Während der ersten Messreihe wurden mehrere Logs über Zeiträume zwischen 12 und 18 Stunden aufgezeichnet. Geloggt wurde ca. einmal pro Minute.[2] Bei dieser Messreihe wurde an der Steuerungseinheit des Moduls-1 Daten über die *serielle-Schnittstelle* ausgelesen und mit der *Serial-Loger*-Software aufgezeichnet.
Diese Software konnte Daten sowohl graphisch darstellen als auch im *csv-Format* speichern. Die erfassten Vorgänge waren der Systemstruktur gemäß auf den Wahrnehmungsbereich der angeschlossenen Einheit begrenzt.
Erfasst wurden ein- und ausgehenden Funksignale, die Sensorwerte, das Wach-/Ruhepotential, der Wach-/Ruhezustand, das Gesundheitspotential, die Lebenszeit, die Sendeleistung und der Schwellenwert des Helligkeitspotentials. Falls die Einheit evolutionär als Empfänger galt, sollten sich die-

2 Im Falle der Aktivierung der Aktuatoren verlängerte sich die Loggpause bis zu Beendigung der Aktuatoraktion. Sämtliche Messwerte wurden bis zum nächsten Logg zwischengespeichert und dann gemeinsam ausgegeben.

se Schwellenwerte ändern. Um diese Änderungen zu erfassen, wurde der Schwellenwert des Helligkeitspotentials als Beispiel geloggt. Wie nach der Loggphase festgestellt werden musste, existierte ein Fehler in der Übermittlung der Schwellenwerte, der dann in der dritten Versuchsphase behoben wurde.

Phase drei - Ergänzungen

Der Fehler in der Übermittlung der Schwellenwerte während der Evolutionsphase[3] konnte trotz intensiver Suche erst nach Abschluss der Loggphase gefunden werden (siehe Anhang 8.2 Progamme). Ein vollständiges neues Logg konnte nicht mehr durchgeführt werden, da das Versuchssystem aufgrund der extremen winterlichen Witterungsbedingungen und dem darauffolgenden Befall durch Insekten nicht mehr zuverlässig funktionierte.

Die Überprüfung des korrigierten Evoultionsmodus mit funktionierender Schwellenwertübermittlung wurde daher in einem getrennten Versuchsaufbau vorgenommen. In diesem Versuchsaufbau wurde eine Steuerungseinheit mit dem korrigierten Programm versehen und als *krank* bzw. *evolutionärer Empfänger* deklariert. Eine zweite Einheit diente als *evolutionärer Schwellenwertspender* und sendete in kurzen Abständen ein evolutionsauslösendes Signal und neue Schwellenwerte. Die empfangende Einheit konnte die neuen Schwellenwerte nun richtig mit ihren vorhandenen Schwellenwerten verrechnen.

Das Ergebnis ist in der Versuchauswertung dokumentiert.

3 Die zu sendenden Schwellenwerte bewegten sich außerhalb der zum Senden deklarierten byte-Variable. Diese kann nur mit Werten zwischen 0-255 definiert werden. Werte außerhalb dieses Bereiches führten zu einem Überlauf der Variable und damit zu unkontrollierbaren Werten. Das Problem wurde durch die Formatierung der Schwellenwerte auf den für die Variable definierten Bereich behoben.

3.6 Versuchsauswertung

In diesem Kapitel wird der Nachweis über die Funktionalität einzelner Prinzipien und Methoden des Versuchssystems geführt. Zu diesem Zweck werden signifikante Ausschnitte der Loggdateien dargestellt und interpretiert.
In einem ersten Schritt musste die Funktionalität der Grundprinzipien, auf denen die adresslose, dezentrale Kommunikation basiert, nachgewiesen werden. In den darauffolgenden Schritten wurden dann weitere Systemprinzipien und Verhaltensweisen auf ihre Funktionstüchtigkeit überprüft.

Die Loggs wurden in zwei verschiedenen Varianten erfasst.
In der ersten Variante wurden Daten über einen längeren Zeitraum an einer einzelnen Steuerungseinheit aufgezeichnet (siehe Abbildung 101). In der zweiten Variante wurden Daten parallel an zwei Steuerungseinheiten aufgezeichnet. Diese Einheiten befanden sich in entgegengesetzten Bereichen des Versuchsgebäudes (siehe Abbildung 102)

Variante Einzellogg
Überprüfte Prinzipien und Methoden
- Autonome Arbeit
- Ereignisbasiertes Systemverhalten
- Regelbasiertes Systemverhalten
- Synchronisierte Verhaltensweisen
- Redundate Signale
- Minimalinformationen
- Dynamische Sendeleistungsregulierung
- Apellmodus[1]
- Evolutionsmodus[2]
- Mutation

Variante Doppellogg
Überprüfte Prinzipien und Methoden
- Adresslose Netzwerkkommunikation
- Physikalische Reichweitenbeschränkung
- Individuelle Beeinflussung
- Globale Informationsverbreitung

Nicht aus Logfiles ablesbare Erkenntnisse
Skalierbarkeit
- Hardwareredundanz
- Emergentes Verhalten

3.6.1 Grundaufbau des Versuchs

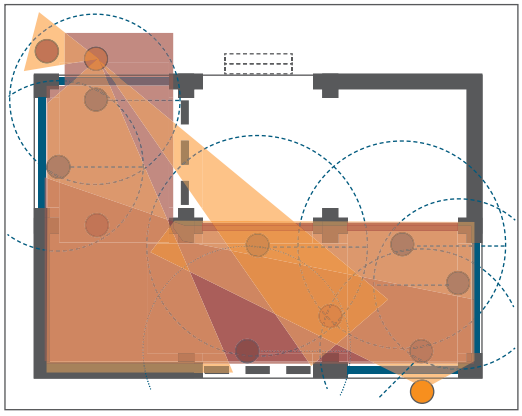


Abbildung 101:
maximale Reichweiten bei niedriger Luftfeuchtigkeit

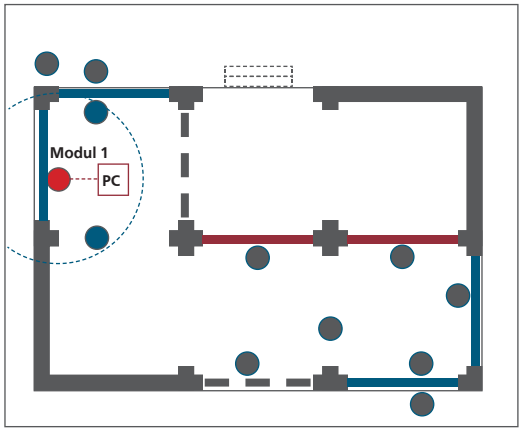


Abbildung 102:
Messaufbau Einzellogg

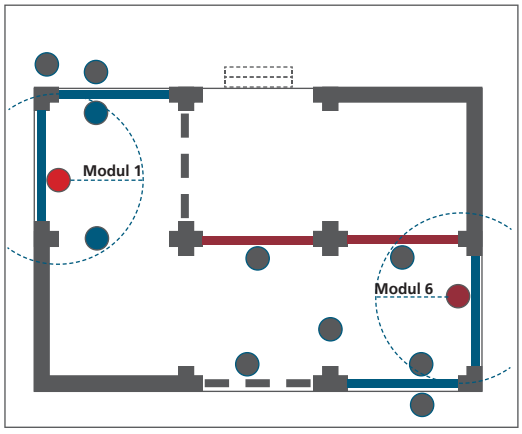


Abbildung103:
Messaufbau Doppellogg

1 Der Apellmodus wurde nur in einer vereinfachten Version getestet. Im Versuch wurde einzig der Weck- bzw. Synchronisationsvorgang umgesetzt. Der Zählapell wurde aufgrund der zu geringen Einheitenanzahl eingespart.

2 Aufgrund der fehlenden Wärmedämmung und Heizung waren alle Einheiten *krank*. Um den Evolutionsmodus dennoch testen zu können wurden die sie über Voreinstellungen auf *gesund* und *krank* gesetzt.

3.6.2 Graphen des Loggfiles: loggerBleiben_20101207_000005_0.ods

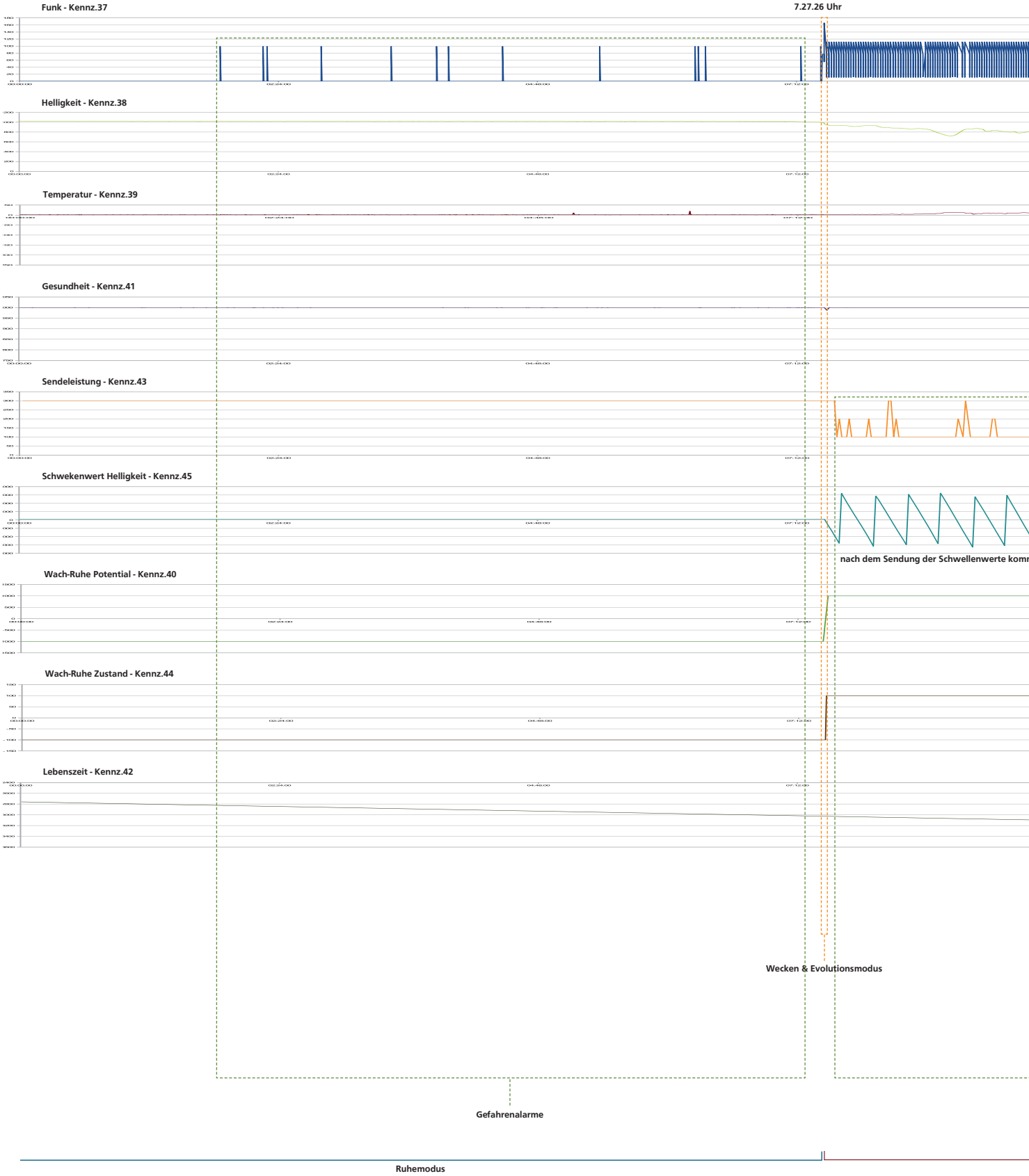
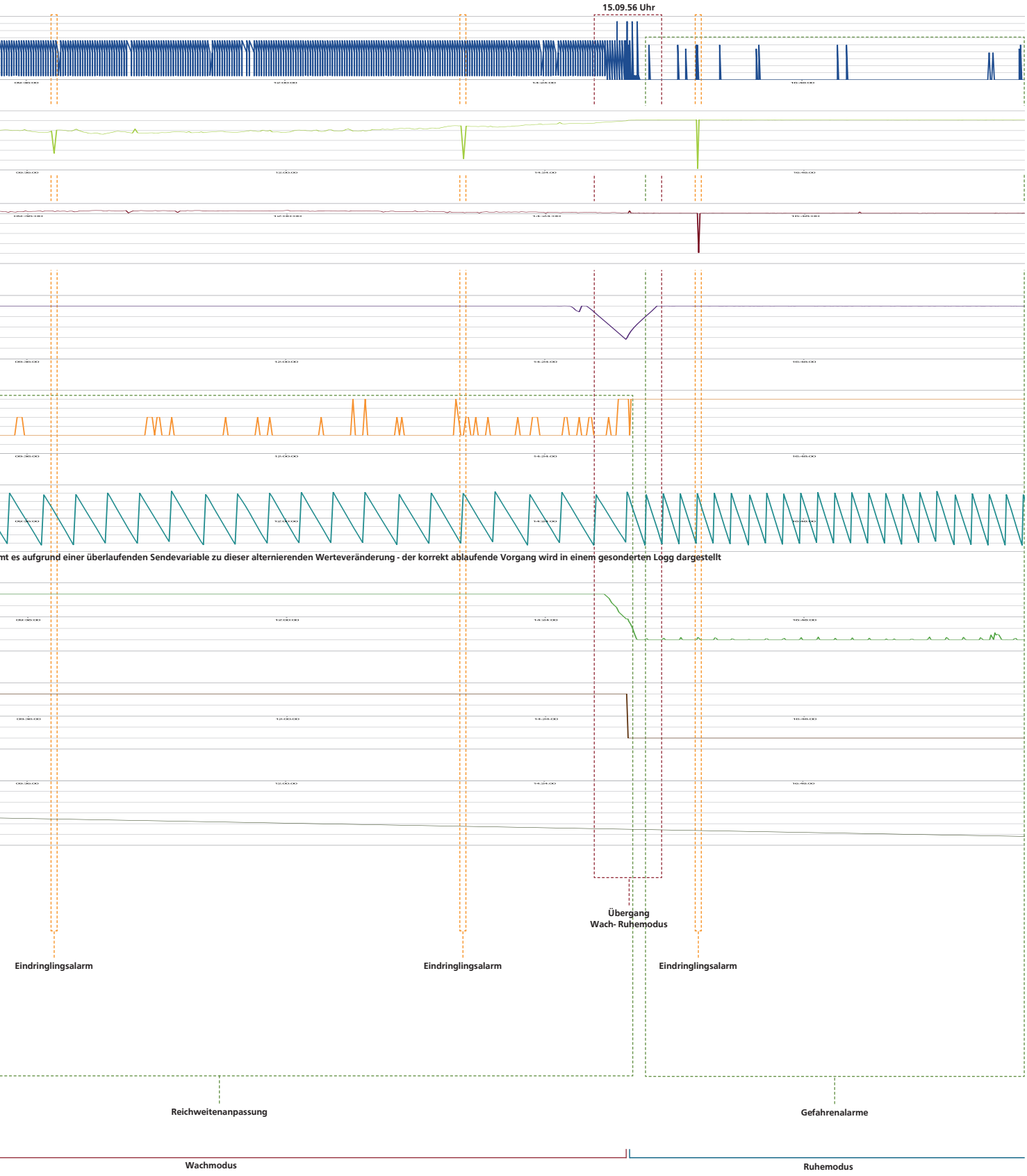


Abbildung 104:
Graph Loggfile loggerBleiben_20101207_000005_0.ods


15.09.56 Uhr
mt es aufgrund einer überlaufenden Sendevariable zu dieser alternierenden Werteveränderung - der korrekt ablaufende Vorgang wird in einem gesonderten Logg dargestellt
Übergang
Wach- Ruhemodus
Eindringlingsalarm
Eindringlingsalarm
Eindringlingsalarm
Reichweitenanpassung
Gefahrenalarme
Wachmodus
Ruhemodus

3.6.3 Logfiles und Interpretation

Das in den Graphen dargestellte Logfile *loggerBleiben_20101207_000005_0.ods* wird hier anhand von zeitlichen Ausschnitten interpretiert (siehe Abbildung 100). Es wurden neun verschiedene Messwerttypen erfasst. Die einzelnen Messwerttypen sind über ihre Kennzeichnung differenzierbar:

Kennzeichnung		Wertebereiche
37	Funk:	Helligkeitssensor:
	empfangene Werte	0 (max. Helligkeit) - 1023 (max. Dunkelheit)
	11 Anfrageping (Reichweite)	
	12 Antwortping (Reichweite)	Temperatursensor:
	20 zu dunkel (Mobileinheit)	0 (+2°C) bis 1023 (155°C) d.h. 0,15°C/pro Einheit
	21 zu hell (Mobileinheit)	Temp. unter +2°C wurden als Wert 0 angezeigt.
	30 zu kalt (Mobileinheit)	
	31 zu warm (Mobileinheit)	Sendeleistung:
	33 Person im Innenraum	3 Abstufungen, wurden zur besseren graphischen
	44 Mobileinheit in Reichweite	Darstellung angezeigt mit den Werten:
	55 Apell Aufruf	100, 200, 300.
	66 Schwellenwerte	
	77 Außenwerte innerhalb Toleranzen	Wach-/Ruhe-Potential: -100.000 (max. Ruhepot.) bis 100.000 (max. Wachpot.)
	88 Windgefahr	
	99 Eindringlingsalarm	Wach-/Ruhe-Zustand:
	gesendete Werte[1]	-100 (Ruhezustand) oder 100 (Wachzstand)
	111 Anfrageping (Reichweite)	
	112 Antwortping (Reichweite)	Lebenzeit:
	155 Apell Antwort	17280000 millisek. + Startwert millisek.
	166 Schwellenwerte	Dies entspricht einer max. Lebenszeit von ca. zwei
38	Helligkeitspotemtial	Tagen. Werte wurden zur besseren graphischen
39	Temperaturpotential	Darstellung mit 100.000 dividiert geloggt.
40	Wach-/Ruhe-Potential	
41	Gesundheit	Schwellenwert Helligkeit (swMHo):
42	Lebenszeit	stabil auf 960 dieser Wert soll ausschließlich wäh-
43	Sendeleistung	rend des Evolutionsmodus leicht modofiziert wer-
44	Wach-/Ruhe-Zustand	den. Dieser Werte wurden zur besseren graphischen
45	Schwellenwert Helligkeit	Darstellung mit 10 dividiert geloggt.

1 Die von der Einheit selber gesendeten Signale werden mit einer 1 vor dem Signalwert aufgezeichnet um sie von empfangenen Signalen zu unterscheiden. Die 166 entspricht also dem Signal 66 und ist mit der vorangestellten 1 als ausgehend gekennzeichnet.

Parallellogs an Modul-1 und Modul-6

Die Erfassung lokal begrenzter Systembeeinflussung und der Nachweis einer globalen Signalverbreitung erforderte zwei parallele Loggs an weit auseinanderliegenden Punkten. Die Datenerfassung erfolgte an den Modulen 1 und 6 (siehe Abbildung 105).

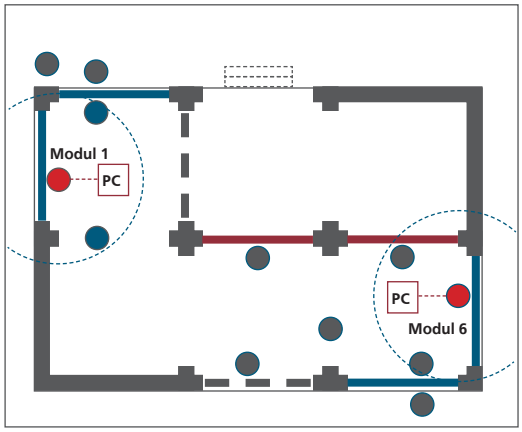


Abbildung 105:
Messaufbau Doppellog

Als Logrechner dienten zwei Notebooks, die die serielle Datenausgabe der Module aufzeichneten. Es war nicht möglich, die Systemzeit der beiden Notebooks exakt zu synchronisieren. Ebensowenig ließ sich die Logsoftware auf beiden Rechnern exakt synchron starten. Des Weiteren unterlagen die zeitlichen Abstände der einzelnen Logs einer Varianz. Das Programm wurde beispielweise durch die Aktivierung der Servos oder Leds in seinem Ablauf gebremst. War dies der Fall, konnte sie den Logvorgang erst nach Beendingung der Aktuatorsteuerung vornehmen.
Hieraus ergeben sich in der Aufzeichung Zeitdifferenzen zwischen den Logfiles der beiden Rechner. Die Logfiles repräsentieren daher nicht exakt den tatsächlichen zeitlichen Ablauf.

Doppellog 1 - Signalweiterleitung

Anhand dieser Logs wurde die globale Singnalverarbeitung untersucht. Die äußere Personendetektionseinheit sendete ein Eindringlingsalarmsignal, das unter anderem von der Steuerungseinheit des Modul-1 empfangen wurde. Einheiten, die das Alarmsignal empfingen, leiteten es weiter, bis es auch die Steuerungseinheit des Modul-6 erreichte (siehe Abbildung 106). Die nördliche äußere Sensoreinheit war mit einem Windsensor ausgestattet. Zum Zeitpunkt des Loggs sandte diese Einheit ein Windgefahrensignal aus. Dieses Signal breitete sich ebenfalls global anhand der Weiterleitung von Einheit zu Einheit aus.

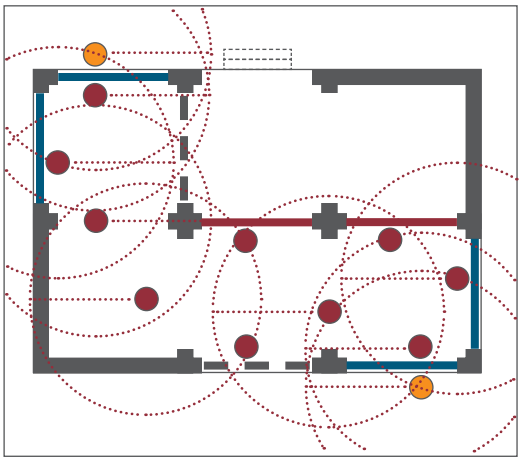


Abbildung 106:
Gefahrenmodus - Ausbreitung

Interpretation:
Die beiden von entgegengesetzen Seiten ausgehenden Gefahrensignale wurden von beiden Einheiten empfangen. Die beiden loggenden Einheiten befanden sich in weit auseinanderliegenden Bereichen des Gebäudes. Hieraus lässt sich ableiten, dass beide Signale global weitergegeben wurden.

Zeitdifferenz und unterschiedliche Logganzahl liegen innerhalb der max. Loggauflösung.

Modul-1
loggerBleiben_20101105_123340_0.ods

Modul-6
loggerBleiben_20101105_123346_0.ods

Kennz.	Wert	Uhrzeit	- Signalweiterleitung		Kennz.	Wert	Uhrzeit
			- Eindringlingsalarm - Windgefahr	- Eindringlingsalarm -Windgefahr			
41	969	13:03:57			41	1000	13:03:52
40	1000	13:03:57			40	1000	13:03:52
37	77	13:03:57			37	77	13:03:52
37	33	13:03:57			37	33	13:03:52
37	99	13:03:57	Eindringlingsgefahr	Eindringlingsgefahr	37	99	13:03:52
37	88	13:03:57	Windgefahr	Windgefahr	37	88	13:03:52
44	100	13:03:57			44	100	13:03:52
45	960	13:03:57			45	960	13:03:52
43	100	13:03:57			43	100	13:03:52
38	852	13:03:57			38	562	13:03:52

Doppellog 2 - lokale Begrenzung Nord-West

Anhand dieser Logs wurde die lokal begrenzte und die individuelle Beeinflussung geprüft. Die Mobileinheit wurde zu diesem Zweck im nord-westlichen Bereich des Versuchgebäudes eingesetzt (siehe Abbildung 107). Die Mobileinheit steht stellvertretend für alle denkbaren Ereignisse, deren Einfluss lokal beschränkt bleiben sollen.

Interpretation:
Der Einfluss der Mobileinheit blieb lokal auf den Bereich um das Modul 1 begrenzt. Das Person-im-Innenraumsignal wurde hingegen global verbreitet.
Zeitdifferenz und unterschiedliche Loganzahl liegen innerhalb der max. Logauflösung.

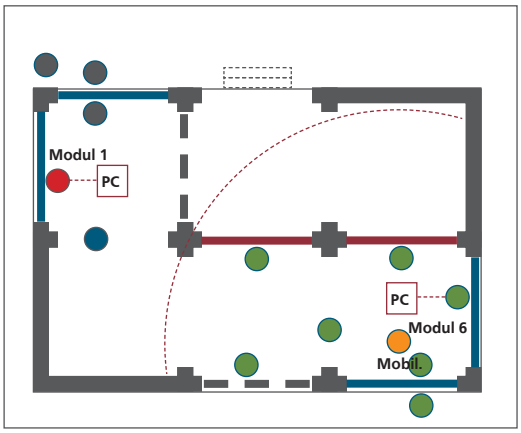


Abbildung 107:
Lokaler Einfluss - nord-westlicher Bereich

Modul-1
loggerBleiben_20101105_123340_0.ods

Kennz.	Wert	Uhrzeit	
41	914	12:43:37	
40	996	12:43:37	
37	77	12:43:37	
44	100	12:43:37	
45	960	12:43:37	
43	100	12:43:37	
38	924	12:43:37	Helligkeit: dunkel
39	49	12:43:37	(Klappen offen)
41	904	12:43:45	
40	1000	12:43:45	
37	77	12:43:45	
37	21	12:43:45	zu hell (Mobil)
37	33	12:43:45	Person im Innenraum
44	100	12:43:45	
45	960	12:43:45	
43	100	12:43:45	
38	314	12:43:45	Helligkeit: hell
39	50	12:43:45	(Klappen schließen)

Modul-6
loggerBleiben_20101105_123346_0.ods

Kennz.	Wert	Uhrzeit	
41	652	12:43:38	
40	998	12:43:38	
37	77	12:43:38	
37	11	12:43:38	
37	12	12:43:38	
37	111	12:43:38	
37	112	12:43:38	
44	100	12:43:38	
45	960	12:43:38	
43	100	12:43:38	
			Helligkeit: dunkel
38	900	12:43:38	
39	48	12:43:38	
41	644	12:43:42	
40	1000	12:43:42	
37	77	12:43:42	keine Signale von
44	100	12:43:42	Mobileinheit
45	960	12:43:42	
43	100	12:43:42	
			Helligkeit: identisch
38	900	12:43:42	
39	49	12:43:42	
41	635	12:43:46	
40	1000	12:43:46	
37	77	12:43:46	
37	33	12:43:46	Person im Inneraum
44	100	12:43:46	
45	960	12:43:46	
43	100	12:43:46	
38	899	12:43:46	Helligkeit: identisch
39	48	12:43:46	

- lokale Begrenzung

Normalbetrieb — Normalbetrieb

Einfluss Mobileinheit — keine Beeinflussung

Doppellog 3 - lokale Begrenzung Süd-Ost

Anhand dieser Logs wurde die lokal begrenzte Beeinflussung und die individuelle Beeinflussung geprüft. Die Mobileinheit wurde zu diesem Zweck im süd-östlichen Bereich des Versuchsgebäudes eingesetzt (siehe Abbildung 108).

Interpretation:
Der Einfluss der Mobileinheit blieb lokal auf den Bereich um das Modul 6 begrenzt.
Zeitdifferenz und unterschiedliche Loganzahl liegen innerhalb der max. Logauflösung.

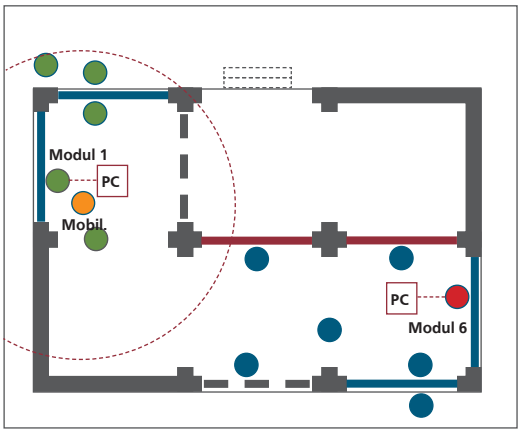


Abbildung 108:
Lokaler Einfluss - süd-östlicher Bereich

Modul-1
loggerBleiben_20101105_123340_0.ods

Kennz.	Wert	Uhrzeit		
			Normalbetrieb	
41	1000	12:39:08		
40	997	12:39:08		
37	77	12:39:08		
37	12	12:39:08		
37	111	12:39:08		
44	100	12:39:08		
45	960	12:39:08		
43	100	12:39:08		
38	857	12:39:08	——	Helligkeit: dunkel
39	50	12:39:08		(Klappen offen)
			keine Beeinflussung	
41	1000	12:39:11		
40	999	12:39:11		
37	77	12:39:11		
44	100	12:39:11		
45	960	12:39:11		
43	100	12:39:11		
38	859	12:39:11	——	Helligkeit: identisch
39	49	12:39:11		

Modul-6
loggerBleiben_20101105_123346_0.ods

Kennz.	Wert	Uhrzeit		
			Normalbetrieb	
41	1000	12:39:02		
40	996	12:39:02		
37	77	12:39:02		
37	11	12:39:02		
37	33	12:39:02		
37	112	12:39:02		
44	100	12:39:02		
45	960	12:39:02		
43	100	12:39:02		
38	876	12:39:02	——	Helligkeit: dunkel
39	46	12:39:02		
			Einfluss Mobileinheit	
41	1000	12:39:09		
40	998	12:39:09		
37	77	12:39:09		
37	11	12:39:09		
37	21	12:39:09	——	Mobileinheit: zu hell
37	112	12:39:09		
44	100	12:39:09		
45	960	12:39:09		
43	100	12:39:09		
38	322	12:39:09	——	Helligkeit: hell
39	48	12:39:09		(Klappen schließen)

Doppellog 4 - lokale Begrenzung zentral

Anhand dieser Logs wurde die lokal begrenzte Beeinflussung geprüft und die individuelle Beeinflussung geprüft. Die Mobileinheit wurde zu diesem Zweck im zentralen Bereich des Versuchsgebäudes eingesetzt (siehe Abbildung 109).

Interpretation:
Der Einfluss der Mobileinheit wirkte sich aufgrund der zentralen Position auf beide loggenden Einheiten aus.
Zeitdifferenz und unterschiedliche Logganzahl liegen innerhalb der max. Loggauflösung.

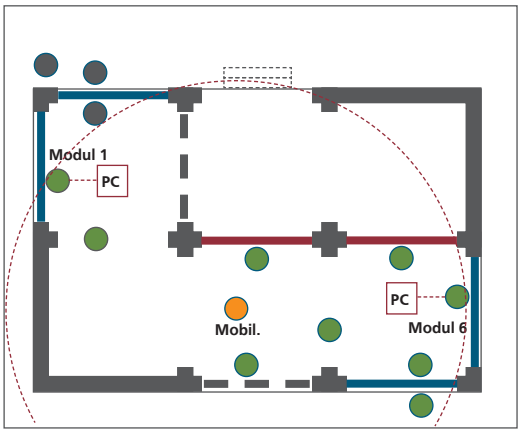


Abbildung 109:
Lokaler Einfluss - Zentraler Bereich

Modul-1
loggerBleiben_20101105_123340_0.ods

Kennz.	Wert	Uhrzeit	
40	997	12:38:24	
37	77	12:38:24	
37	11	12:38:24	
37	12	12:38:24	
37	21	12:38:24	zu hell (Mobil)
37	112	12:38:24	
44	100	12:38:24	
45	960	12:38:24	
43	100	12:38:24	
38	60	12:38:24	
39	50	12:38:24	

Einfluss Mobileinheit

Modul-6
„loggerBleiben_20101105_123346_0.ods

Kennz.	Wert	Uhrzeit	
41	1000	12:38:23	
40	997	12:38:23	
37	77	12:38:23	
37	11	12:38:23	
37	12	12:38:23	
37	21	12:38:23	zu hell (Mobil)
37	111	12:38:23	
37	112	12:38:23	
44	100	12:38:23	
45	960	12:38:23	
43	100	12:38:23	
38	-864	12:38:23	
39	48	12:38:23	

Einfluss Mobileinheit

Einzellog
loggerBleiben_20101207_000005_0.ods

Kennz.	Wert	Uhrzeit	Vorgänge	Beschreibung	
			- Ruhe-/Wachmodus - Appell - Evolution - Reichweitenanp.	Werte befinden sich im Ruhebereich	
42	-23027	07:26:26			
40	-692	07:26:26		Wach-/Ruhepotential:	noch im negativen Bereich
37	77	07:26:26		äußere Sensoreinheit:	erster pos. Helligkeitswert
44	-100	07:26:26		Wach-/Ruhezustand:	Ruhezustand
45	960	07:26:26			
43	300	07:26:26			
38	992	07:26:26			
39	3	07:26:26			
41	1000	07:26:26		Helligkeit und Temperatur steigen und übersteigen die Wachschwelle der äußeren Sensoreinheiten	
42	-23027	07:26:53			
40	-312	07:26:53		Wach-/Ruhepotential:	ansteigend
37	77	07:26:53		äußere Sensoreinheit:	positiver Helligkeitswert
44	-100	07:26:53		Wach-/Ruhezustand:	Ruhezustand
45	960	07:26:53			
43	300	07:26:53			
38	990	07:26:53			
39	3	07:26:53			
41	1000	07:26:53			
				Wechsel in Wachmodus, Appell, Evolution	
42	-23027	07:27:26			
40	36	07:27:26		Wach-/Ruhepotential:	im positiven Bereich
37	77	07:27:26		äußere Sensoreinheit:	positiver Helligkeitswert
37	55	07:27:26		Appell:	Appellsignal empfangen
37	155	07:27:26		Appell:	Appellsignal gesendet
37	166	07:27:26		Evolution:	Schwellenwert gesendet
44	100	07:27:26		Wach-/Ruhezustand:	Wachzustand
45	960	07:27:26		Schwellenwert Hell.:	Startwert
43	300	07:27:26		Sendeleistung:	maximum
38	963	07:27:26		Helligkeit:	ansteigend
39	3	07:27:26		Temperatur:	gleichbleibend
41	999	07:27:26			
				Reichweitenanpassung	
42	-23028	07:28:45			
40	998	07:28:45		Wach-/Ruhepotential:	im positiven Bereich
37	77	07:28:45		äußere Sensoreinheit:	positiver Helligkeitswert
37	11	07:28:45		Reichweite:	Anfrageping empfangen
37	12	07:28:45		Reichweite:	Antwortping empfangen
37	111	07:28:45		Reichweite:	Anfrageping gesendet
37	112	07:28:45		Reichweite:	Antwortping gesendet
44	100	07:28:45		Wach-/Ruhezustand:	Wachzustand
45	-3719	07:28:45		Schwellenwert Hell.:	verändert (Evo. fehlerhaft)
43	100	07:28:45		Sendeleistung:	maximum

Kennz.	Wert	Uhrzeit	Vorgänge	Beschreibung	
			- Eindringlingsalarm		
				Normaler Tagesbetrieb im Wachmodus	
42	-23112	09:49:36			
40	999	09:49:36			
37	77	09:49:36			
37	11	09:49:36			
37	12	09:49:36			
37	111	09:49:36			
37	112	09:49:36			
44	100	09:49:36			
45	13840	09:49:36			
43	100	09:49:36			
38	789	09:49:36	———	Helligkeit:	mittel
39	13	09:49:36	———	Temperatur:	ca. 4°C
41	1000	09:49:36			
				Eindringlingsalarm; Modifizierung des Helligkeits- und des Temperaturpotentials führt zur Schließung der Klappen und somit zur Sicherung des Gebäudes	
42	-23113	09:51:02			
40	999	09:51:02			
37	77	09:51:02			
37	11	09:51:02			
37	12	09:51:02			
37	99	09:51:02	———	Eindringlingsalarm:	redundante Signale! empfangen u. weitergeleitet
37	111	09:51:02			
37	112	09:51:02		(Empfang & Weiterleitung nicht getrennt geloggt)	
44	100	09:51:02			
45	8888	09:51:02			
43	100	09:51:02			
38	336	09:51:02	———	Helligkeit:	gesteigert
39	10	09:51:02	———	Temperatur:	gesenkt
41	1000	09:51:02			
				Normalisierung der Werte	
42	-23113	09:52:28			
40	999	09:52:28			
37	77	09:52:28			
37	11	09:52:28			
37	12	09:52:28			
37	111	09:52:28			
37	112	09:52:28			
44	100	09:52:28			
45	3758	09:52:28			
43	100	09:52:28			
38	812	09:52:28	———	Helligekeit:	mittel
39	13	09:52:28	———	Temperatur:	ca. 4°C
41	1000	09:52:28			

Kennz.	Wert	Uhrzeit	Vorgänge	Beschreibung	
			- Eindringlingsalarm		
				Normaler Betrieb im Ruhemodus	
42	-23248	13:37:21			
40	999	13:37:21			
37	77	13:37:21			
37	11	13:37:21			
37	12	13:37:21			
37	111	13:37:21			
37	112	13:37:21			
44	100	13:37:21			
45	-30039	13:37:21			
43	100	13:37:21			
38	898	13:37:21	———	Helligkeit:	dunkel
39	7	13:37:21	———	Temperatur:	ca. 3,5°C
41	1000	13:37:21			
				Eindringlingsalarm; Modifizierung des Helligkeits- und des Temperaturpotentials führt zur Schließung der Klappen und somit zur Sicherung des Gebäudes	
42	-23249	13:38:47			
40	999	13:38:47			
37	77	13:38:47			
37	11	13:38:47			
37	12	13:38:47			
37	99	13:38:47	———	Eindringlingsalarm:	redundante Signale! empfangen u. weitergeleitet
37	111	13:38:47			
37	112	13:38:47		(Empfang u. Weiterleitung nicht getrennt geloggt)	
44	100	13:38:47			
45	30011	13:38:47			
43	100	13:38:47			
38	225	13:38:47	———	Helligkeit:	gesteigert
39	4	13:38:47	———	Temperatur:	gesenkt
41	1000	13:38:47			
				Normalisierung der Werte	
42	-23250	13:40:13			
40	1000	13:40:13			
37	77	13:40:13			
37	11	13:40:13			
37	12	13:40:13			
37	111	13:40:13			
37	112	13:40:13			
44	100	13:40:13			
45	25511	13:40:13			
43	200	13:40:13			
38	893	13:40:13	———	Helligkeit:	dunkel
39	6	13:40:13	———	Temperatur:	ca. 2,15°C
41	1000	13:40:13			

Kennz.	Wert	Uhrzeit	Vorgänge	Beschreibung
			- Wach-/Ruhemodus	
				Werte befinden sich im Wachbereich
42	-23303	15:09:08		
40	-90	15:09:08		Wach-/Ruhepotential: bereits im negativen Bereich
37	0	15:09:08		äußere Sensoreinheit: keine Informationssignale
37	11	15:09:08		
37	12	15:09:08		
37	111	15:09:08		
37	112	15:09:08		
44	100	15:09:08		Wach-/Ruhezustand: noch im Wachzustand
45	-28738	15:09:08		
43	300	15:09:08		
38	998	15:09:08		Helligkeit: dunkel
39	4	15:09:08		Temperatur: 2,6°C
41	842	15:09:08		
				Wechsel in den Ruhemodus; Da ein Wechsel in den Wachmodus sichtbar ist, sank das Wach-/Ruhepotential bereits unter die Schwelle von -100 und stieg dann noch einmal kurz an.
42	-23304	15:09:56		
40	-77	15:09:56		Wach-/Ruhepotential: kurzer Anstieg
37	0	15:09:56		
37	11	15:09:56		
37	12	15:09:56		
37	111	15:09:56		
37	112	15:09:56		
44	-100	15:09:56		Wach-/Ruhezustand: Ruhezustand
45	31671	15:09:56		
43	300	15:09:56		
38	1007	15:09:56		Helligkeit: sehr dunkel
39	3	15:09:56		Temperatur: 2,45°C
41	852	15:09:56		
				Ruhemodus
42	-23304	15:10:34		
40	-200	15:10:34		Wach- Ruhepotential: kurzer Anstieg
37	0	15:10:34		Der kurze WR- Potential Anstieg weckte das System
37	11	15:10:34		noch einmal und löste die Appell- und Evolutionsmo-
37	12	15:10:35		di erneut aus.
37	55	15:10:35		Apell: Start
37	66	15:10:35		Evolution: Schwellenwerte empfangen
37	99	15:10:35		Eindringlingsalarm: Fehlalarm! Signal nicht
37	166	15:10:35		redundant = keine Reaktion
37	112	15:10:35		
44	-100	15:10:35		Wach-/Ruhezustand: Ruhezustand
45	28563	15:10:35		
43	300	15:10:35		
38	1012	15:10:35		Helligkeit: sehr dunkel
39	14	15:10:35		Temperatur: 4,1°C
				(Sensor variiert bis zu 15 Einheiten)

Kennz.	Wert	Uhrzeit	Vorgänge	Beschreibung	
			- Eindringlingsalarm		
				Normaler Betrieb im Ruhemodus	
42	-23327	15:48:16			
40	-994	15:48:16			
37	0	15:48:16			
44	-100	15:48:16			
45	-30342	15:48:16			
43	300	15:48:16			
38	1022	15:48:16	———	Helligkeit:	dunkel
39	2	15:48:16	———	Temperatur:	ca. 2,3°C
41	999	15:48:16			
				Eindringlingsalarm; Modifizierung des Helligkeits- und des Temperaturpotentials führt zur Schließung der Klappen und somit zur Sicherung des Gebäudes	
42	-23327	15:49:00			
40	-894	15:49:00			
37	0	15:49:00			
37	99	15:49:00	———	Eindringlingsalarm:	redundante Signale! empfangen u. weitergeleitet
44	-100	15:49:00			
45	29915	15:49:00		(Empfang u. Weiterleitung nicht getrennt geloggt)	
43	300	15:49:00			
38	24	15:49:00	———	Helligkeit:	gesteigert
39	-196	15:49:00	———	Temperatur:	gesenkt
41	1000	15:49:00			
				Normalisierung der Werte	
42	-23328	15:49:43			
40	-944	15:49:43			
37	0	15:49:43			
37	99	15:49:43			
44	-100	15:49:43			
45	25099	15:49:43			
43	300	15:49:43			
38	1022	15:49:43	———	Helligkeit:	dunkel
39	1	15:49:43	———	Temperatur:	ca. 2,15°C
41	1000	15:49:43			

Logfile vom 3.11.2010
loggerBleiben_20101103_143638_0.ods

Kennz.	Wert	Uhrzeit	Vorgänge	Beschreibung
			- Windgefahr	
				Normaler Betrieb im Ruhemodus
41	-13927	15:44:59		
40	47	15:44:59		
37	0	15:44:59		
44	100	15:44:59		
45	960	15:44:59		
43	100	15:44:59		
38	978	15:44:59	———	Helligkeit: dunkel
39	43	15:44:59	———	Temperatur: ca. 8,45°C
				Windgefahr
41	-13937	15:45:16		
40	37	15:45:16		
37	0	15:45:16		
37	88	15:45:16	———	Windgefahr: redundante Signale! empfangen u. weitergeleitet
44	100	15:45:16		(Empfang u. Weiterleitung nicht getrennt geloggt)
45	960	15:45:16		
43	100	15:45:16		
38	19	15:45:16	———	Helligkeit: gesteigert
39	-29	15:45:16	———	Temperatur: gesenkt
[...]				
				Normaler Betrieb im Ruhemodus
41	-13704	15:48:20		
40	-401	15:48:20		
37	0	15:48:20		
37	88	15:48:20	———	Windgefahr: noch keine redundanten Signale
44	-100	15:48:20		
45	960	15:48:20		
43	100	15:48:20	———	Helligkeit: dunkel
38	998	15:48:20	———	Temperatur: ca. 8,45°C
39	43	15:48:21		
				Windgefahr
41	-13695	15:48:23		
40	-410	15:48:23		
37	0	15:48:23		
37	88	15:48:23	———	Windgefahr: redundante Signale! empfangen u. weitergeleitet
44	-100	15:48:23		(Empfang u. Weiterleitung nicht getrennt geloggt)
45	960	15:48:23		
43	100	15:48:23		
38	1	15:48:23	———	Helligkeit: gesteigert
39	-55	15:48:23	———	Temperatur: gesenkt

Evolutionäre Schwellenwertveränderung

Aufgrund des beschädigten Versuchssystems wurde der Prozess der evolutionären Schwellenwertübertragung in einem getrennten Versuchsaufbau geloggt (siehe Abbildung 110).

Die übertragenen Schwellenwerte wurde durch10 dividiert, um mit einem Byte übertragen werden zu können.

Um 16.41.20 ereignete sich ein Übetragungsfehler (Wert 0), woraufhin die Werte stark mutierten (siehe Abbildungen 107,108).

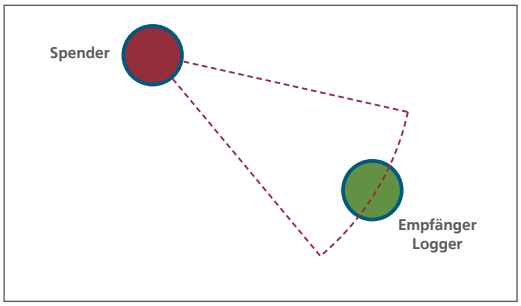


Abbildung 110:
Versuchsaufbau Evolution

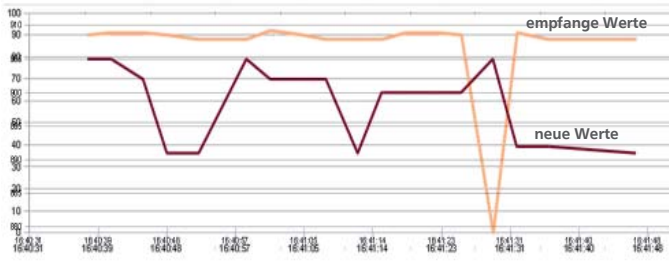


Abbildung 111:
evolutionäre Änderung Schwellenwert Hell. oben

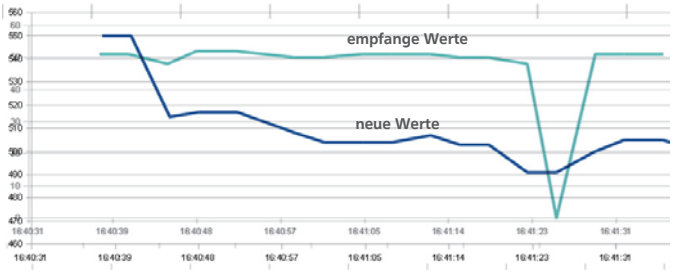


Abbildung 112:
evolutionäre Änderung Schwellenwert Hell. unten

loggerBleiben_20110527_164036_0.ods

Kennz.	Wert	Uhrzeit	Vorgänge	Beschreibung
			- Evolution korrigiert	Schwellenwertempfang
42	91	16:40:45		Empfang Schwellen. Hell oben.: 91 x 10 = 910
40	525	16:40:45		
37	0	16:40:45		
37	55	16:40:45		Appellsignal: Synchronisation starten
37	66	16:40:45		Evolution: neue Schwellenwerte
37	111	16:40:45		
44	902	16:40:45		Schwellen. Hell. oben: neu verrechnet
45	515	16:40:45		Schwellen. Hell. unten: neu verrechnet
43	300	16:40:45		
38	146	16:40:45		
39	165	16:40:45		
41	48	16:40:45		Empfang Schwellen. Hell. unten: 48 x 10 = 480
				Schwellenwertempfang
42	90	16:40:48		Empfang Schwellen. Hell oben.: 90 x 10 = 900
40	630	16:40:48		
37	0	16:40:48		
37	55	16:40:48		Appellsignal
37	66	16:40:48		Evolution: neue Schwellenwerte
37	111	16:40:48		
44	891	16:40:48		Schwellen. Hell. oben: neu verrechnet
45	517	16:40:48		Schwellen. Hell. unten: neu verrechnet
43	300	16:40:48		
38	147	16:40:48		
39	162	16:40:48		
41	52	16:40:48		Empfang Schwellen. Hell. unten: 52 x 10 = 520

3.6.4 Auswertung

Betrachtung zweier Einheiten

Adresslose Netzwerkkommunikation

Zunächst musste überprüft werden, ob es grundsätzlich möglich ist, eine kontrollierbare Netzwerkkommunikation ohne Adressierung der Einheiten aufzubauen.

Der Lösungsansatz bestand in einer physikalischen Begrenzung der Informationsausbreitung, die im Ergebnis in einer räumlichen Reichweitenbegrenzung resultierte. Diese ersetzte die sonst übliche softwarebasierte Begrenzung der Informationsausbreitung. Eine softwarebasierte Begrenzung der Informationsausbreitung ist konzeptbedingt auf eine Adressierung der Einheiten angewiesen und konnte somit in ADRRM-Systemen nicht angewendet werden.

Das Versuchssystem beruhte daher auf einer adresslosen Komunikation mit physikalischer Ausbreitungsbegrenzung.

Wie aus den *Doppelloggs* ersichtlich ist, war das System in der Lage, ohne adressierbare Einheiten räumlich kontrolliert zu kommunizieren. Das System war dabei sowohl zu lokal begrenzter Kommunikation in der Lage (siehe *Doppelloggs 2,3 und 4*) als auch zu globaler Kommunikation (*siehe Doppellogg 1*).

Physikalische Reichweitenbeschränkung

Die *Doppelloggs 2,3 und* 4 zeigen, dass diese physikalische Reichweitenbegrenzung in ihrer Auflösung fein genug abgestimmt werden konnte, um sie zur Steuerung von Gebäuden einzusetzen. Der Einflussbereich der Mobileinheit umfasste etwa einen Radius von fünf Metern, wurde aber von physischen Hindernissen wie Mauerwerk blockiert.

Im Versuch ist aus den *Doppelloggs 2 und 3* ersichtlich, dass der Einflussbereich auf einen Bereich von ca. fünf mal zweieinhalb Metern begrenzt war. Die Mobileinheit sendete dabei auf mittlerer Leistung, der Einflussbereich ließe sich also sowohl erweitern als auch reduzieren.

Der Einflussbereich könnte etwa ein Einzelbüro umfassen oder den Bereich zweier Einzelarbeitsplätze. In einem Sitzungsraum, auf Verkehrsflächen oder in Sozialräumen würden sich die Bereiche der Mobileinheiten überschneiden. Im Versuchssystem wären verschiedene Anforderungen der Mobileinheiten über die Potentiale verrechnet worden. In umfangreicheren ADRRM-Systemen könnten weitere Abstimmungsmechanismen vorgesehen werden (siehe 3.3.3 Organisationsstrategien Staatsbildender Insekten und 3.4.2 Systemstruktur). Da es nur eine Mobileinheit gab, ist beispielsweise in dem Logfile *loggerBleiben_20101105_123346_0.ods* nur eine Verrechung des Einflusses der Mobileinheit mit dem Helligkeitspotential zu sehen, das um 12.39.09 Uhr von 876 auf einen Wert von 322 sinkt. Wäre eine zweite Mobileinheit mit hebendem Einfluss vor Ort gewesen, hätten sich ihre Einflüsse in etwa aufgehoben.

Individuelle Beeinflussung

Eine individuelle Beeinflussung der Umgebung ist notwendig, falls ein Nutzer Wünsche hat, die von den systemeigenen Umgebungseinstellungen abweichen. In zentralen Systemen erfordert dies die Identifikation des Nutzers und die Bestimmung seiner Position, um von der zentralen Steuerung aus die standortrelevante Infrastruktur zu regeln. Dezentrale Systeme sind auf ein solches Vorgehen nicht angewiesen. In dezentralen Systemen kann der Einfluss des Nutzers direkt vor Ort erfasst und verarbeitet werden. Dies kann beispielweise durch fest montierte Sensoren geschehen.

In dem Versuchsaufbau erfasste eine Sensoreinheit Personen im Innenraum und meldete dies an benachbarte Einheiten weiter. Die Anwesenheit von Personen verhinderte den Wechsel des Systems in den Ruhezustand. Dieser Einfluss wurde festgeschrieben, da ein Nutzer während seiner Arbeit ein *waches*, dynamisch arbeitendes System benötigt. Allerdings wäre eine Nutzerwunscherkennung über fest installierte Taster oder Sensoren sehr aufwendig und unkomfortabel. Eine Integration des Nutzers in das Steuerungskollektiv ist dem vorzuziehen. Der Nutzer kann anhand seiner individuellen Mobileinheit Wünsche an das System übermitteln. Dies kann von Fall zu Fall aber auch automatisiert geschehen, so dass die Umgebung seines jeweiligen Standortes seinen Wünschen gemäß geregelt wird.

In dem Versuchsaufbau wurde beispielsweise der Wunsch nach einer Reduzierung der Helligkeit übermittelt. Dies wurde in den *Doppelloggs 2,3 und 4* aufgezeichnet. Da in ADRRM-Systemen grundsätzlich keine Anweisungen übermittelt werden, er-

folgt die Einflussnahme des Nutzers einzig über die Einspeisung potentialverändernder Signale. Die Signale der Mobileinheit suggerierten hierzu den in Reichweite befindlichen Einheiten eine extreme Helligkeit, woraufhin diese mit dem Schließen der Verschattungsklappen reagierten und die Helligkeit reduziert wurde. Wie aus den *Doppelloggs 2,3 und 4* ersichtlich wird, geschieht dies nur in der näheren Umgebung der Mobileinheit, entferntere Bereiche bleiben unbeeinflusst. Im praktischen Einsatz würde dies beispielweise bedeuten, dass der Sonnenschutz der Fenster eines Büros aktiviert wird, während das Nachbarbüro nicht beeinflusst wird.

Globale Informationsverbreitung

Die Einheiten des Versuchssystems sind aufgrund ihrer eingeschränkten Funkreichweite nicht in der Lage, Informationen unmittelbar systemweit zu verbreiten. Dies ist in Gefahrensituationen aber unbedingt erforderlich. Um dieser Anforderung gerecht zu werden, ist es den Einheiten möglich, Informationen von einer Einheit zur anderen weiterzugeben. Die Einheiten, die das Gefahrensignal empfangen, leiten das Gefahrensignal wiederum an die benachbarten Einheiten weiter. Um hierbei eine permanente Selbstaktivierung bzw. einen Rundlauf der Signale zu vermeiden, leiteten die Einheiten nicht zweimal hintereinander das gleiche Signal weiter.[1]

Im *Doppellogg 1* ist zu sehen, wie sich zwei Gefahrensignale von entgegengesetzten Seiten aus über das System verbreiteten. Die loggenden Einheiten befanden sich ebenfalls in entgegengesetzten Bereichen des Versuchsgebäudes. In beiden Loggs sind die beiden unterschiedlichen Gefahrensignale aufgezeichnet, dies war nur durch die Weiterleitung der Gefahrensignale möglich. Die globale Informationsverbreitung über die Weiterleitung von Einheit zu Einheit hatte also funktioniert.

Betrachtung einer einzelnen Einheit

Ereignisbasiertes Systemverhalten

Die im Versuchssystem eingesetzten Einheiten besaßen keine Uhrenmodule, die ihnen eine Bestimmung der *Echtzeit*[2] ermöglicht hätten. Sie konnten zwar die Zeit seit dem Programmstart messen, nach einem Stromausfall oder Neustart verloren die Einheiten diese Zeitinformation allerdings. Eine *Echtzeitbestimmung* wurde allerdings auch nicht benötigt, da es sich bei ADRRM-Systemen um ereignisbasierte Systeme handelt. Ereignisbasierte Systeme kennen keine festen Zeiten, zu denen sie bestimmte Aktivitäten tätigen sollen. Die Aktivitäten ereignisbasierter Systeme beruhen ausschließlich auf Regeln. In diesen Regeln wird festgelegt, aufgrund welcher Umgebungsaktivitäten Systemaktivitäten ausgelöst werden sollen. Der Vorteil dieser regelbasierten Aktivitäten ist die Unabhängigkeit von voreingestellten Terminen und eine flexible Reaktionsfähigkeit auf unerwartete Ereignisse.

Der Wechsel vom Wach- in den Ruhemodus und umgekehrt unterliegt beispielsweise ausschließlich den Umgebungsfaktoren. Anhand des Logfiles *loggerBleiben_20101207_000005_0.ods* ist zu erkennen, wie das System im Zeitraum von 7.27.26 bis 7.28.45 Uhr von dem Ruhemodus in den Wachmodus wechselt. Die loggende Einheit empfing das Informationssignal einer außenliegenden Sensoreinheit. Dieses Signal übermittelte ihr, dass die Außenhelligkeit nun in dem Wachbereich der äußeren Sensoreinheit lag. Aufgrund dieses Signales wurden die innenliegenden Einheiten geweckt. Das Informationssignal bewirkt einen Anstieg des Wach-/Ruhepotenials bei den empfangenden Einheiten. Das Potential stieg über die Wachschwellen und die Einheiten wechselten daraufhin in den Apell- und den Evolutionsmodus, um dann in den Wachmodus überzugehen. Um 15.09.56 wechselt die loggende Einheit aufgrund nachlassender Helligkeit wieder in den Ruhemodus.

Die Einheit war also in der Lage, ereignisbasiert aus dem Ruhemodus in den Wachmodus zu wechseln und ebenso ereignisbasiert wieder in den Ruhemo-

1 Alle Gefahrensignale werden redundant gesendet. Dies bedeutet, dass nach dem Empfang eines Gefahrensignals dieses mehrere Male hintereinnander weitergeleitet wird. Dieser Vorgang wird als eine Signalweiterleitung betrachtet, da er nur einmal pro empfangenen Gefahrensignal erfolgt. Empfängt diese Einheit noch einmal das gleiche Gefahrensignal, wird es nicht weitergeleitet. Erst nachdem sie ein anderes Signal empfangen hat, kann sie wieder ein Gefahrensignal des gleichen Typs weiterleiten. In diesem Fall wird davon ausgegangen, dass es sich um ein neues Gefahrensignal handelt, nicht um einen Rundlauf des vorhergehenden.

2 Die sogenannte *Echtzeit* meint die geltende Uhrzeit vor Ort. Im Gegensatz zur *Systemzeit*, die sich nach anderen Bedingungen, wie beispielsweise dem Systemstart richten kann. Eine Bestimmung der *Echtzeit* erfordert im Falle einer nicht dauerhaften Stromversogung zusätzliche Echtzeitmodule mit eigener ausfallsicherer Stromversogung.

dus zurückzukehren.
Kurz zuvor wurde allerdings ein überraschendes Ereignis protokolliert: ein zweiter Appell- und Evolutionsmodus. Eine Umgebungsveränderung führte zu einem erneuten kurzfristigen Wiederanstieg des Wach-/Ruhepotentials über die Wachschwelle. Daraufhin wechselte das System noch einmal in den Wachmodus. Dies war ein völlig regelkonformer Vorgang. Bei der Platzierung der Regel wurde davon ausgegangen, dass der Verlauf der Helligkeits- und Temperaturwerte über den Tag dermaßen gleichmäßig erfolgen würde, dass ausschließlich morgens ein Wechsel vom Ruhe- in den Wachmodus möglich wäre. Dies war anscheinend nicht der Fall und lieferte ein anschauliches Beispiel für das Versagen einer Regel oder ihrer Schwellenwerteinstellungen.

Regelbasiertes Systemverhalten

Die Verhaltenweisen zentraler Systeme basieren auf virtuellen Umgebungsmodellen, aus denen die benötigten Steuerungsvorgänge errechnet werden. Die Verwendung solcher Modelle setzt zentrale Strukturen voraus und ist daher in dezentralen Systemen nicht möglich.
Dezentrale Systeme sind auf rein regelbasierte Verhaltensweisen angewiesen. Sämtliche in den Loggs ersichtlichen Vorgänge sind dementsprechend regelbasiert. Die Aufgaben eines zentralen Steuerungsmodells wurden im Versuchssystem in Form von Regelsätzen auf das System verteilt. Das System arbeitete somit auf der Basis *Verteilter Intelligenz*. Die zwei grundlegenden Schwierigkeiten bei der Einrichtung regelbasierter Systeme bestehen in der Regelfindung und der korrekten Einstellung der Schwellenwerte dieser Regeln.
Das im Logfile *loggerBleiben_20101207_000005_0.ods* um 15.09.56 zu beobachtende zweite Erwachen an einem Tag spiegelt diese Schwierigkeit gut wider. Entweder war die Regel, den Wechsel des Ruhe- in den Wachmodus abhängig von Umgebungsfaktoren zu machen, ungünstig oder unvollständig oder die Schwellenwerte, die den Zustandswechsel steuern, waren ungünstig gesetzt. Es aber auch möglich, dass es sich um ein außergewöhnliches Ereignis in der Umgebung handelte, das keine Regeländerung rechtfertigt.
Um mit dieser ambivalenten Situation umgehen zu können, benötigt die Einheit wiederum Regeln, die ihr eine Beurteilung der Regeln ermöglichen. Eine Regel könnte beispielsweise in der Beurteilung der Energieeffizienz eines Vorgangs bestehen. Im konkreten Fall ist allerdings festzustellen, dass eine zweite evolutionäre Phase im Grunde keine Nachteile mit sich bringt. Die Regeln können daher bestehen bleiben. Falls eine zweite evolutionäre Phase absolut unerwünscht sein sollte, kann dies einzig über eine neue Regel, beispielsweise einen festgeschriebenen Mindestzeitraum zwischen den evolutionären Phasen, geregelt werden.
Es bleibt festzustellen, dass eine regelbasierte Verhaltenssteuerung extrem flexibel, aber eben auch extrem diffizil einzustellen ist. Um die Verhaltensweisen zu optimieren, sind wiederum Regeln notwendig, die wiederum optimiert werden müssen. In ADRRM-Systemen wird zu diesem Prozess ein evolutionärer Mechanismus eingesetzt, der zur Eliminierung von Regeln führt, die das *Überleben* der Einheit gefährden (siehe Kapitel 3.4.2 Systemstruktur). Regeln, die das *Überleben* der Einheit nicht gefährden, bleiben in diesem Prozess allerdings über längere Zeiträume erhalten.

Synchronisierte Verhaltensweisen

Die Einheiten des Versuchssystems besaßen, wie bereits erwähnt, keine Echtzeituhren. Eine zeitbasierte Synchronisation war ihnen daher nicht möglich. In dezentralen Netzwerk ohne synchronisierte Kommunikation ist eine Synchronisation der Einheiten sogar geradezu schädlich. Falls die Systemeinheiten, ohne auf reservierte Zeitfenster zu warten, einfach zu einem beliebigen Zeitpunkt senden, dürfen sie auf keinen Fall synchron laufen. Wären diese Einheiten synchronisiert, könnte es geschehen, dass sämtliche Einheiten zum gleichen Zeitpunkt senden. Das hätte wiederum zur Folge, dass keine der Einheiten empfangsbereit wäre. Die gesendeten Signale würden völlig ungehört und ohne Reaktionen verhallen. Das System wurde daher mit einem eigens eingerichteten Mechanismus desynchronisiert. Da es aber dennoch einen Vorgang gibt, der eine Synchronisation erfordert, den Zählappell, wurde eine Programmroutine eingerichtet, die eine kurzzeitige Synchronisierung des Systems ermöglicht.
Diese Routine sorgte dafür, dass nach dem Wechsel aus dem Ruhemodus, wie in dem Logfile *loggerBleiben_20101207_000005_0.ods* aufge-

zeichnet, um 07.27.26 ein Synchronisationssignal (37/55 und 37/155) ausgesendet wurde. Dieses Signal sollte eigentlich zu einem Zählappell führen. Aufgrund des relativ kleinen Versuchssystems mit einer sehr begrenzten Anzahl von Einheiten, wurde das Appellsignal einzig zur Synchronisation des Evolutionsmodus genutzt. Der Evolutionsmodus wurde dann direkt anschließend durchgeführt (37/66 und 37/166). Nach Abschluss des evolutionären Schwellenwertaustausches erfolgte dann ein desynchronisierter Start des Wachmodus, der sicherstellte, dass das System bis zum nächsten Appell asynchron lief. Die aktive Desynchronisierung ist auch aufgrund des Phänomens der potentiellen Eigensynchronisation von Netzwerken notwendig.[3]
Eine partiell synchronisierte Kooperation in einem normalerweise asynchronen Netzwerk ist, wie im Log zu sehen, möglich, muss hiernach aber unbedingt über eine aktive Desynchronisation wieder aufgehoben werden.

Redundante Signale

Sensoren minderer Qualität, verrauschte Funkfrequenzen und eine Kommunikation ohne Empfangsbestätigung führen zu fehlerhaften Signalen. Eine Maßnahme zur Sicherung der Signaleindeutigkeit innerhalb verrauschter Kommunikation besteht auf Seiten des Senders in der Übermittlung redundanter Signale. Auf Seiten des Empfängers erfolgt eine Reaktion dazu kohärent einzig auf redundante Signale. In dem Versuchssystem wirken die Signale anderer Einheiten ausschließlich beeinflussend. Die Signale bestehen niemals aus direkten Anweisungen. Ein empfangenes Signal hebt oder senkt Potentiale, die allerdings auch anderen Einflüssen unterworfen sind. Dies betrifft in dem Versuchssystem vor allem die Helligkeits- und Temperaturpotentiale, die als Basis für die Steuerung der Klappen dienen.
Mehrfach gesendete bzw. redundante Signale des gleichen Typs werden im Logg nur als ein Wert ausgegeben und sind daher nur aufgrund der Reaktionen der Einheit von einzelnen nicht redundanten Signalen differenzierbar.
Im Logfile *loggerBleiben_20101207_000005_0.ods* ist um 15.45.16 Uhr ein Gefahrensignal (37/99) zu erkennen. Daraufhin ist eine drastische Reduzierung des Helligkeitspotentials von 978 auf 19 sowie des Temperaturpotentials von 43 auf -19 sichtbar. Es handelte sich somit um ein redundantes Signal.
Zum Zeitpunkt 15.48.20 wurde ebenfalls ein Gefahrensignal (37/99) empfangen, es ist aber keine signifikante Änderung der Helligkeits- und Temperaturpotentiale sichtbar. Bei diesem Signal handelte es sich um ein einzelnes, nicht redundantes Signal.
Um 15.48.23 wurde wiederum ein Gefahrensignal (37/99) angezeigt und es ließ sich eine Potentialveränderung erkennen. Es handelte sich demnach wiederum um ein redundantes Signal.

Minimalinformationen

Das Versuchssystem kommuniziert anhand sehr reduzierter Informationen. In den Logfiles wurde allerdings ausschließlich das erste Byte eines Signals aufgezeichnet. Dies ist insofern ausreichend, da außer zur Übertragung der Schwellenwerte im Evolutionsmodus nur das erste Byte zu Kommunikation genutzt wurde. Das bedeutet, dass 99,9 Prozent der Kommunikation über ein Byte abgewickelt wurde, das die gesamte Information enthielt. Diese Kommunikation kann damit berechtigt als Kommunikation mit minimaler Informationsbreite bezeichnet werden.
Diese minimale Informationsbreite bringt den Vorteil einer minimalen Funkbelastung mit sich. Die Einheiten des Versuchssystems sendeten mit einer maximalen Leistung von 2,4mW und benötigten zur Übermittlung eines Signals ca. 22ms.[4] Ein Signal hatte eine Gesamtlänge von 22 Byte. Fünfzehn Byte wurden für den Aufbau der Kommunikation benötigt und nur sieben Byte tragen das eigentliche Signal. Sechs der sieben Byte wurden dabei ausschließlich während der Schwellenwertübermittlung genutzt. Die genaue Anzahl der Signale pro Minute kann aufgrund der groben zeitlichen Loggauflösung bedauerlicherweise nicht zuverlässig ermittelt werden. Die Funkbelastung ist jedoch in Umgebungen, in denen Wlan-Netze (max. 1 Watt Sendeleistung), Funktelefone (max. 0,25 Watt Sendeleistung) und Mobiltelefone (max. 1-2 Watt Sendeleistung) im Einsatz sind, von irrelevantem Ausmaß.

Dynamische Sendeleistungsregulierung

Die Reichweite elektromagnetischer Wellen im Bereich der eingesetzten Frequenzen ist von der Sen-

3 Vgl. METZLER, R. et al.: Synchronisation neuronaler Netzwerke, in: Physical Review E, 62, 2000.

4 Sparkfun Datenblatt: Single chip 2.4 GHz Transceiver nRF2401A. (http://www.sparkfun.com/datasheets/IC/nRF2401A.pdf), 27. Mai. 2011.

deleistung, der Antenne, der Luftfeuchtigkeit und eventuell vorhandenen physikalischen Hindernissen abhängig. Um unter diesen Umständen eine gleich bleibende Signalreichweite zu erhalten, ist eine dynamische Regulierung der Sendeleistung notwendig, die sich den teils ebenfalls dynamischen Umgebungsfaktoren anpasst.
In dem Logfile *loggerBleiben_20101207_000005_0.ods* wurde um 7.26.53 die Sendeleistung mit voller Leistung auf Stufe 3(43/300) verzeichnet. In der Ruhephase war die Sendeleistung auf der höchsten Stufe fixiert, da in dieser Zeit auf Aktivitäten weitgehend verzichtet werden sollte und daher keine Reichweitenanpassung stattfinden sollte. Die Sendeleistung wurde im Ruhemodus auf der höchsten Stufe fixiert, um eventuelle Gefahren in jedem Fall kommunizieren zu können. Da Gefahrensignale global verarbeitet werden, spielte die einzig für lokale Verhaltensweisen benötigte räumliche Reichweitenbeschränkung keine Rolle.
Nachdem die Einheit um 7.27.26 Uhr erwachte, startete sie umgehend die Routine zur Sendeleistungsregulierung. Um 7.28.45 wurde aufgezeichnet, dass die Einheit sowohl Anfragepings aussandte (37/111) als auch empfing (37/11). Sie beantwortete die Anfragen ihrerseits mit Antwortpings (37/112) und erhielt ebenfalls Antwortpings anderer Einheiten (37/12). Offensichtlich empfing sie bereits innerhalb des ersten Zeitfensters mindestens zwei Antwortpings verschiedener Einheiten, denn sie senkte ihre Sendeleitung auf die niedrigste Stufe 1 (43/100).

Appellmodus

Der Appellmodus wurde in dem Versuchssystem einzig zur Synchronisierung des Evolutionsmodus eingesetzt.
Das Versuchsgebäude war nicht isoliert und konnte nicht beheizt werden, daher konnten keine Temperaturen innerhalb der Überlebenstoleranzen der Einheiten gehalten werden. In der Folge schalteten alle Einheiten auf den Empfängerstatus. Spendereinheiten waren danach inexistent. Ein Zählappell war daher nicht mehr sinnvoll durchführbar und wurde ausgelassen.
In dem Logfile *loggerBleiben_20101207_000005_0.ods* wurde der Synchronisationsvorgang um 7.27.26 aufgezeichnet. Die Einheit erwachte und sandte ein Appellsignal (37/155) aus, woraufhin sie ein Antwortsignal einer anderen Einheit empfing (37/55). Der Empfang eines Appellsignals bewirkte eine kurze Pause im Programmverlauf, um allen Einheiten die Gelegenheit zu bieten, sich auf den Evolutionsmodus vorzubereiten. Zudem wurde der Countdown des auf zehn Sekunden begrenzten Evolutionszeitraums gestartet. *Kranke* bzw. Schwellenwert empfangende Einheiten hatten nun maximal zehn Sekunden Zeit, neue Schwellenwert zu empfangen (37/66). Erhielten sie innerhalb dieser Zeit keine neuen Werte, brachen sie die evolutionäre Phase ab und mutierten ihre Werte eigenständig.

Evolutionsmodus

Da aufgrund der baulichen Gegebenheiten entweder sämtliche Einheiten simultan auf den Status *krank* gesetzt wurden oder nach Modifikation der Toleranzen auf winterliche Temperaturen und Lichtverhältnisse sämtliche Einheiten *gesund* blieben, konnte ein evolutionärer Werteaustausch nicht stattfinden. Um den Evolutionsmodus dennoch prüfen zu können, wurden drei Einheiten per Programmeingriff auf den *Spenderstatus* und drei auf den *Empfängerstatus* fixiert. Nach der Synchronisierung durch das Appellsignal wurden nun Daten von den Spendern zu den Empfängern übertragen. In der Programmversion, die während der Aufzeichnung des Logfiles *loggerBleiben_20101207_000005_0.ods* eingesetzt wurde, befand sich ein Fehler, der nach Aktivierung des *Evolutionsmodus* um 7.28.45 zu einer alternierenden Veränderung des Schwellenwertes des Helligkeitspotentials führte, die im Graphen gut sichtbar ist. Dies bedeutet dennoch, dass der Evolutionsmodus eingesetzt hat und auch ein Werteaustausch vorgenommen wurde.
Der Programmfehler konnte erst nach Abschluss der Versuchsaufzeichnungen gefunden werden. Da der Versuchsaufbau zu diesem Zeitpunkt zu beschädigt war, um dort ein weiteres Log vorzunehmen, wurde die evolutionäre Schwellenwertübertragung in einem getrennten Versuch überprüft. Hierbei wurden zwei Einheiten eingesetzt, eine als Spender und eine als Empfänger. Wie aus dem Logfile *loggerBleiben_20110527_164036_0.ods* ersichtlich ist, wurde um 16.40.45 und um 16.40.48 zunächst ein Appellsignal gesendet und danach neue Schwellenwerte (41/48 und 42/91 - bzw. 41/52 und 42/90). Es ist zu beachten, dass die gesendeten Schwellenwerte durch 10 dividiert wurden, um sie auf einen Wert zu reduzieren, der über ein Byte gesendet werden konnte. Die empfangenen Werte mussten dementsprechend wieder mit 10 multipliziert werden,

um die ursprüglichen Werte wiederherzustellen (die Werte waren daher nach der Wiederherstellung gerundet, siehe Abschnitt Mutation). Die wiederhergestellten empfangenen Schwellenwerte wurden daraufhin mit den alten Schwellenwerten der Einheit zu neuen Schwellenwerten verrechnet(44/902 und 45/515 bzw. 44/891 und 45/517).

Mutation

Um das System in seiner evolutionären Entwicklung dynamisch zu halten, ist es notwendig, eine Vielfalt an Schwellenwerten aufrechtzuerhalten. Zu diesem Zweck müssen permanent neue Schwellenwerte generiert werden, ansonsten erfolgt eine Verarmung der im System befindlichen Schwellenwerte. In dem Loggfile *loggerBleiben_20110527_164036_0.ods* sind zwei von drei Maßnahmen sichtbar, die Schwellenwerte mutieren lassen und so neue Werte in das System einspeisen. Nicht zu erkennen ist, dass die Schwellenwerte vor der Übertragung durch die Addition eines Zufallswerts verändert werden (siehe 7.2 Programme). Ist dies geschehen, werden sie durch 10 dividiert, woraus sich gerundete Werte ergeben. Bei diesem Vorgang werden die Schwellenwerte somit erneut verändert. Dies ist an den übertragenen Werten des Loggfiles *loggerBleiben_20110527_164036_0.ods* zu sehen. Die übertragenen Schwellenwerte (41/48 und 42/91) korrspondieren mit den vorhandenen Schwellenwerten (45/515 und 44/902), mit denen sie verrechnet wurden (siehe auch 7.2 Progamme).
Im Graphen zu dem Loggfile *loggerBleiben_20110527_164036_0.ods* ist eine Fehlübertragung zu sehen. Es wurde einmal der Wert Null übermittelt. Diese Fehlübertragung führte zu einer starken Mutation der Schwellenwerte. Dies ist im Sinne der Generierung neuer Werte durchaus als Vorteil anzusehen. Ein Fehlübertragung wird deshalb absichtlich nicht unterbunden.

Versuchsergebnisse
(nicht aus den Loggfiles ersichtlich)

Autonome Arbeit

Der Versuch zeigte zunächst einmal, dass die einzelnen Einheiten in der Lage sind, autonom zu arbeiten. Dies lässt sich aus den Loggs nicht direkt ablesen, da die Betätigung der Aktuatoren nicht erfasst wurde.
Es ist aus dem Graphen allerdings ersichtlich, dass die geloggte Einheit mit ihren eigenen Sensoren Daten erfasste und auf Basis dieser Sensorwerte ihre Potentiale veränderte. Auf Grundlage dieser Potentialveränderungen erfolgten dann Aktuatoraktionen. Dies weist die Fähigkeit zu autonomer Arbeit indirekt nach. Die autonome Arbeit der Einheiten ist zudem filmisch dokumentiert unter: *http://vimeo.com/22550950 .*

Skalierbarkeit

Dezentrale Systeme sind einfach skalierbar. Im Falle einer redundanten Aufgabenverteilung betrifft diese Skalierbarkeit auch die Wegnahme von Systemeinheiten. Dies kann aufgrund eines Systemrückbaus oder durch Ausfälle einzelner Einheiten geschehen. Im Falle des Versuchssystems trat eine Rückwärtsskalierung durch Ausfälle auf. Das System verfügte über zwei außenliegende Sensoreinheiten, die redundant die äußeren Umgebungswerte in das System einspeisten. Der Ausfall der nördlichen Einheit hatte keine negativen Auswirkungen auf das Systemverhalten, da die innen angebrachten Einheiten auch von der verbleibenden südlichen Sensoreinheit zuverlässig geweckt und informiert wurden.
Des Weiteren fielen im Laufe des Versuchs zwei Steuerungseinheiten aus und mussten ersetzt werden. Der Ausfall der Einheiten hatte ebenfalls keine Auswirkungen auf das verbleibende System, da die Kommunikationswege redundant ausgelegt waren. Nach dem Austausch der unter anderem durch echte *Bugs*[5] beschädigten Einheiten arbeitete das System ohne Probleme weiter. Dies kann als Beispiel einer problemlosen Systemerweiterung bzw. Skalierung betrachtet werden.

Hardwareredundanz

Die eben genannten Beispiele belegen die Relevanz redundanter Hardware. Dies zeigte sich auch an einem Beispiel fehlender Hardware-Redundanz. Von den ehemals beabsichtigten zwei Windsensoren wurde im Versuch nur einer montiert. Dies stellte sich als Fehler heraus. Der einzige Windsensor des Systems fiel nach dem Bruch einer Lötstelle aus. Die-

5 Als Bug werden sowohl Software als auch Hardwarefehler bezeichnet, seit Grace Hopper einen imaginären Käfer zeichnete, den sie für falsche Daten verantwortlich machte, 1944 fand sie dann tatsächlich einen toten Käfer in einem beschädigten Relais.
Vgl. MARX, Christy: Grace Hopper. The First Woman to Program the First Computer in the United States. New York 2004, S. 43-44.

ser Ausfall wurde weder rechtzeitig bemerkt, noch konnte er durch einen redundanten Sensor kompensiert werden. Windgefahr wurde nun nicht mehr signalisiert und die Klappen schlossen sich bei Wind nicht mehr. In der Folge wurden die Klappen durch Windböen beschädigt.
Hardware sollte, wie der Versuch zeigte, in dezentralen Systemen immer redundant ausgelegt werden, da ein Ausfall einzelner Elemente nicht zentral angezeigt wird. Hardwareausfälle müssen bis zu einer Systeminspektion über redundante Elemente kompensierbar sein.

Emergentes Verhalten

Das emergente Systemverhalten ist aus den Loggdateien nicht direkt ersichtlich, da das Verhalten einer oder zweier Einheiten keinen Überblick in das Gesamtsystemverhalten geben kann. Inwieweit der Versuch ein gobal erscheinendes Verhalten aufwies, ist daher anhand der Loggfiles nicht ersichtlich.
Während der Versuchsphase konnte beobachtet werden, dass das installierte System die Klappen des Versuchsgebäudes steuerte, zwischen dem Ruhe- und dem Wachmodus wechselte und auf Gefahren und auf Umgebungsveränderungen reagierte. Die dabei auftretenden globalen Verhaltensweisen waren an keiner Stelle des Systems als solche implementiert. Das System könnte daher als *emergentes System in einer sehr schwachen Form* betrachtet werden (3.4.1 Systemtheoretische Grundlagen).
Aufgrund des geringen Umfangs des Versuchssystem mit maximal vierzehn Einheiten war das globale Systemverhalten grob vorhersehbar, im Detail allerdings nicht.
Es war beispielweise vorhersehbar, dass sich die Verschattungsklappen bei höherer Sonneneinstrahlung schließen würden. Wann und wo sich welche Klappen schließen würden, war hingegen nicht vorhersehbar. Da dies auch von den Lichtverhältnissen über den Tag hinweg abhängig war und diese für jede Klappen anders ausfielen, konnten die Faktoren, von denen das globale Verhalten abhing, vom Beobachter nicht nachvollzogen werden.
Ein weiteres Beispiel fand sich im Regelbetrieb während des Wachmodus. Selbst wenn bekannt war, aufgrund welcher Bedingungen die Klappen geöffnet und geschlossen wurden, war es nicht vorhersehbar, wann genau welche Klappen aktiviert wurden. Die große Anzahl der Abhängigkeiten machte es unmöglich, präzise Voraussagen zu treffen.
Ebenso verhielt es sich mit dem Wechsel vom Ruhe- in den Wachmodus. Dieser Wechsel geschah synchron, aber ereignisbasiert. Eine Vorhersage, wann es zu diesem Wechsel kommen würde, war nicht exakt möglich, da er außer von den Umgebungsbedingungen auch von der Ereignissen des Vortags abhängig war. Wann das System erwachen würde und welche Einheit den Wechsel initiieren würde, konnte nicht exakt vorhergesagt werden.
Auch der globale evolutionäre Prozess verlief koordiniert und sinnvoll, es war allerdings nicht vorhersehbar, welche Einheiten als Schwellenwertspender und welche als Empfänger auftreten würden.
Das System verhielt sich somit global *deterministisch chaotisch* aber sinnvoll und erschien damit (*schwach-) emergent*.

3.6.5 Tabelle Versuchauswertung

Prinzip - Verhaltensweise	im Versuch	funktionsfähig	Loggfile	Kommentar
autonome Arbeit	ja	ja	loggerBleiben_20101207_000005_0.ods gesamtes Logg	
adresslose Kommunikation	ja	ja	loggerBleiben_20101207_000005_0.ods gesamtes Logg	
Minimalinformationskommunikation	ja	ja	loggerBleiben_20101207_000005_0.ods gesamtes Logg	
verteilte Intelligenz	ja	ja	nicht in Loggs sichtbar	
physikalische Reichweitenbeschränkung	ja	ja	loggerBleiben_20101207_000005_0.ods 7.27.26	
regelbasiertes Verhalten	ja	ja	loggerBleiben_20101207_000005_0.ods gesamtes Logg	
ereignisbasiertes Verhalten	ja	ja	loggerBleiben_20101207_000005_0.ods 7.27.26 - 15.09.56	
fähig zu emergentem Verhalten	ja	-	nicht in Loggs sichtbar	
individuelle Beeinflussbarkeit	ja	ja	Doppelloggs 2,3,4	
lokal beschränkte Beeinflussbarkeit	ja	ja	Doppelloggs 2,3,4	
globale Informationsverbreitung	ja	ja	Doppellogg 1	
synchroner/asynchroner Betrieb	ja	ja	loggerBleiben_20101207_000005_0.ods 7.27.26	
Skalierbarkeit	ja	ja	nicht in Loggs sichtbar	
Signalredundanz	ja	ja	loggerBleiben_20101103_143638_0.ods 15.45.16 - 15.48.20	
Hardwareredundanz	ja	ja	nicht in Loggs sichtbar	
sinnvolle evolutionäre Entwicklung	nein	?	-	im Versuchsaufbau nicht prüfbar
dynamische Sendeleistungsanpassung	ja	ja	loggerBleiben_20101207_000005_0.ods 7.27.26 - 7.28.45	
Personendetektion Innenraum	ja	ja	loggerBleiben_20101105_123340_0.ods	
Eindringlingsalarm	ja	ja	loggerBleiben_20101207_000005_0.ods 9.51.02	
Windalarm	ja	ja	loggerBleiben_20101103_143638_0.ods 15.45.16	
Appellmodus	(ja)	ja	loggerBleiben_20101207_000005_0.ods 7.27.26	
Evolutionsmodus	ja	ja	loggerBleiben_20101207_000005_0.ods 7.27.26	
Akustisches Feedback	ja		Film: http://vimeo.com/22550950 -min6.45	
Wechsel Ruhe-/Wachmodus	ja		loggerBleiben_20101207_000005_0.ods 7.27.26	
Wechsel Wach-/Ruhemodus	ja		loggerBleiben_20101207_000005_0.ods 15.09.56	

4 Wirtschaftlichkeit

4.1 Wirtschaftlichkeit

Der Aspekt der Wirtschaftlichkeit wurde in die These bewusst nicht aufgenommen, denn Wirtschaftlichkeit war keine Prämisse bei der Entwicklung dieses Systems. Dennoch sollen in diesem Kapitel einige die Wirtschaftlichkeit betreffenden Fakten genannt werden.

4.2 Systemkosten

Eine belastbare Schätzung der Kosten eines serienreifen Systems ist in dem aktuellen Entwicklungsstand des Versuchssystems nicht möglich.
Die einzigen belastbaren Kosten für eine Systemeinheit resultieren aus den Kosten der Prototypeneinheiten des Versuchssystems. Diese Einheiten enthalten Material im Wert von ca. 40 Euro[1] bei einer Montagezeit von ca. einer Stunde. Sie konsumieren maximal 0.23 Watt und erfordern keine aktive Kühlung.

4.3 Bauliche Infrastruktur

ADRRM-Systeme führen im Bereich der baulichen Infrastruktur zu einer Kostenersparnis im Vergleich zu zentral gesteuerten Systemen. Zentrale Systeme benötigen Steuerungsserver. Diese Server produzieren je nach Systemgröße Abwärme, die abgeführt werden muss. Ab einer bestimmten Größe müssen Server in klimatisierten Räumen untergebracht werden.
Da ADRRM-Systeme ohne zentrale Steuerung arbeiten, entfallen die Kosten für die Server, die Severräume, die Kühlung oder Klimatisierung der Server und der Teil der elektrischen Energie, der in den Servern zu Abwärme gewandelt wird (siehe Abbildung 113).

Der prozentuale Anteil der Einsparungen lässt sich in dem aktuellen Entwicklungsstand nicht beziffern.

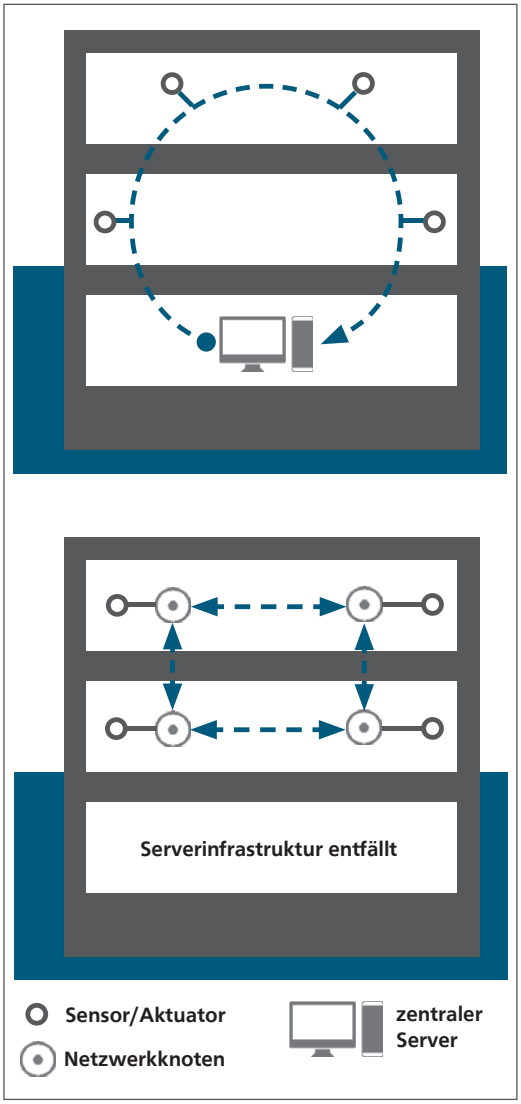


Abbildung113:
Infrastruktur zentraler und dezentraler Syteme

4.4 Detaillierte Steuerungsauflösung

Die Steuerung zentraler Systeme basiert auf Modellen. Modelle sind per Definition immer abstrahierte Abbilder der Wirklichkeit. Regelungsmodelle besitzen demnach einen geringeren Detaillierungsgrad als die Umgebung, die sie abbilden; sie sind abstrahierte Modelle der Wirklichkeit. Der Grad der Abstraktion bestimmt in zentralen Systemen die Qualität der Regelung. Je detaillierter die Modelle, desto

1 Matrerialkosten ADRRM Versuchseinheit

Platine	4,30 €
Arduino Mini Pro	13,77 €
Transceiver	15,96 €
Sensor Helligkeit	1,09 €
Sensor Temperatur	2,00 €
Kleinteile ca.	3,00 €
gesamt	40,12 €

Preise *Sparkfun* und *Watterott Elektronic* Februar 2011

optimierter fällt die Steuerung aus. Allerdings steigt mit der Detaillierung der Modelle auch der Rechenaufwand, der zu ihrer Generierung benötigt wird. Des Weiteren werden die Modelle mit zunehmender Detaillierung immer komplizierter und Änderungen dieser Modelle werden immer aufwendiger. Dies führt zu kostenintensiven Anpassungen der Modelle bzw. dem Betrieb auf Basis nicht angepasster Modelle. Die Regelungsmodelle werden immer einen Kompromiss zwischen maximal möglicher Abstraktion und maximal möglicher Detaillierung darstellen. Eine auf solchen Modellen basierende Steuerung muss daher zwangläufig Vorgänge pauschalisieren. Diese Pauschalisierung kann zu suboptimal geregelten Bereichen der Umgebung führen und damit zu Energieverlusten.

Die Regelung durch ADRRM-Systeme basiert nicht auf Modellen. ADRRM-Systeme arbeiten direkt in der zu steuernden Umgebung. Die Systemreaktionen erfolgen direkt vor Ort in Echtzeit und sind damit extrem dynamisch. ADRRM-Systeme müssen nicht an veränderte Umgebungsparameter angepasst werden, sie sind selbstanpassend ausgelegt. Die Kosten für eine Anpassung von Regelungsmodellen entfallen. Der Detallierungsgrad, mit dem ADRRM-Systeme ihre Umgebung steuern, hängt allein von der Anzahl der Einheiten ab. Die Anzahl der Einheiten kann jederzeit an die Erfordernisse angepasst werden. Die Rechenleistung der ADRRM-Systeme dient vollständig der Steuerung. Die Rechenleistung, die bisher zur Generierung eines Steuerungsmodelles diente, kann somit ebenfalls zur Steuerung der Umgebung genutzt werden. In ADRRM-Systemen bedeutet eine Steigerung der Rechenleistung eine Steigerung der Anzahl der Systemeinheiten. In der Folge vergößert sich die Dichte der Einheiten in Bezug zu ihrer Umgebung. Eine höhere Einheitendichte ist gleichzusetzen mit einer detaillierteren Sensorik und diese ermöglicht eine detailliertere Steuerung. ADRRM-Systeme mit vergleichbarer Rechenleistung sind so in der Lage wesentlich detaillierter zu steuern als zentrale Systeme.

ADRRM-Systeme können dynamischer und detaillierter reagieren als zentrale Systeme. Die Regelungsmodelle zentraler Systeme können eingespart werden, ebenso die Anpassung derselben. ADRRM-Systeme können ihre Umgebung folglich wirtschaftlicher steuern als zentrale Modelle.

4.5 Einsatz im Bestand

ADRRM-Systeme eigenen sich aufgrund ihrer geringen Anforderungen an die bauliche Infrastruktur gut für den Einsatz in bestehenden Strukturen. In Altbauten müssen bei der Umrüstung auf ADRRM-Systeme keine Kommunikationsleitungen gelegt werden, da ADRRM-Systeme über Funk kommunizieren. Falls ADRRM-Einheiten auf Basis von Hochspannungsmikrokontrollern aufgebaut werden können, könnten sie direkt an die Verbraucher angeschlossen werden, so dass keine zusätzliche Niederspannungsstromversorgung notwendig wäre. ADRRM-Systeme könnten zudem bestehende Steuerungsnetze ersetzen, ohne die Aktuatoren auswechseln zu müssen. Die Vernetzung und Steuerung würde in diesem Fall von dem ADRRM-Systemen übernommen. Den Aktuatoren gegenüber würden die einzelnen Einheiten das vorher vorhandene Netz simmulieren, indem sie ihnen Befehle in dem Protokoll übermitteln, für das die Aktuatoren ausgelegt wurden. Da in dem ADRRM-Protokoll ausschließlich die Netzwerkkommunikation festgelegt ist, können ADRRM-Einheiten mit den Sensoren und Aktuatoren in diversen Protokollen kommunizieren, ohne dass die ADRRM Netzwerkkommunikation verändert werden muss. ADRRM-Systeme können somit grundsätzlich auch alle bestehenden Gebäudesteuerungssysteme ersetzen.
Auch momentan noch nicht gesteuerte Infrastrukturen können mit ADRRM-Systemen nachgerüstet werden. Dies kann je nach Anwendungsfall über eine Nachrüstung mit Relais, Dimmern oder anderen Steuerungsgeräten, die von den Systemeinheiten angesprochen werden können, geschehen. Auf diese Weise können zum Beispiel bestehende Wegebeleuchtungen automatisiert und somit energieeffizient gesteuert werden.

4.6 Betrieb

ADRRM-Systeme sind im Betrieb wirtschaftlicher als zentrale Systeme, da sie durch ihre selbstanpassende Auslegung zur Inbetriebnahme oder bei Änderung der Umgebungsparameter keinen personellen Einsatz erfordern. Eventuelle Wartungsarbeiten beschränken sich auf einen Austausch defekter Einheiten, der keine Spezialkenntnisse erfordert.

5 Gefahren und Unannehmlichkeiten

5.1 Unannehmlichkeiten

Die von ADRRM-Systemen ausgehenden Unannehmlichkeiten werden hauptsächlich aus der Varianz zwischen Nutzeraktion und Systemreaktion resultieren.

5.1.1 Beeinflussung statt Anweisungen

Der Umgang mit ADRRM-Systemen erfolgt auf andere Weise als der Umgang mit zentral gesteuerten Systemen. Im Falle eines ADRRM-Systems besitzt der Nutzer gegenüber dem System keine Weisungsbefugnis, er ist auf die Beeinflussung des Systems beschränkt. Sein Einflusspotential ist von diversen weiteren Faktoren abhängig, die für eine Systementscheidung mitentscheidend sind.

ADRRM-Systeme reagieren immer mit der energieeffizientesten Aktion. Diese Aktion ist abhängig von einer ganzen Reihe von Umgebungsfaktoren und zurückliegenden Ereignissen, welche sich der Wahrnehmung der Nutzer oft entziehen. Dies kann auf der Nutzerseite zu Irritationen führen, da auf Situationen, die für den Nutzer scheinbar identisch sind, nicht immer identische Systemreaktionen erfolgen. Zudem lässt das System keine gefährlichen, widersprüchlichen oder in energetischer Hinsicht unsinnigen Aktionen zu. Auch in diesen Fällen sind dem Nutzereinfluss Grenzen gesetzt, die für ihn nicht immer nachvollziehbar sein werden. Aus diesen Systemverhaltenweisen resultiert eine teilweise Entmachtung und Bevormundung des Nutzers, welche vom Nutzer als störend empfunden werden könnte. Die hierarchische Positionierung der Systementscheidungen über die Einflussnahme der Nutzer ist allerdings für eine energieeffiziente und sichere Automatisierung unumgänglich.

Es ist anzunehmen, dass die Nutzer nach einiger Zeit die Systemreaktionen einzuschätzen lernen und sie sich an den ausschließlich beeinflussenden Umgang mit dem System gewöhnen. Inwiefern dies zutrifft, muss zu einem späteren Zeitpunkt in einem Einsatz des Systems unter realen Bedingungen eingehender untersucht werden.

5.1.2 Emergente Verhaltensweisen

ADRRM-Systeme entwickeln *emergente Verhaltensweisen.* Das bedeutet, dass ADRRM-Systeme globale Verhaltensweisen entwickeln, die sich aus der Programmierung der Einheiten nicht direkt ableiten lassen. *Emergente Verhaltensweisen* sind daher nicht vorhersehbar (siehe Kapitel 3.4.1 Systemtheoretische Grundlagen). Systeme mit *emergenten Verhaltensweisen* sind sehr anpassungsfähig, da sie keinen vorgegebenen Verhaltensweisen folgen müssen. *Emergente Systeme* können auf diese Weise allerdings auch Problemlösungen entwickeln, die der Nutzer nicht nachvollziehen kann und die sich für ihn eventuell störend auswirken. *Emergentes Systemverhalten* sollte daher gedämpft und begrenzt werden. Dies geschieht über *Nutzer-System-Toleranzengleicheit*, indem die Systemtoleranzen auf Werte begrenzt werden, die sich innerhalb der Behaglichkeitswerte der Nutzer bewegen. Eine Regelung der Umgebung außerhalb unangenehmer Bereiche ist somit ausgeschlossen.
Zudem sollten *emergente Systeme* nicht zur ausschließlichen Steuerung lebenswichtiger Systemkomponenten wie zum Beispiel von Brandmeldeanlagen oder Schließanlagen eingesetzt werden. Aus Sicherheitsgründen sollten hier einfache Systeme mit fixierten voreingestellten Gefahrenabwehrreaktionen verwendet werden.

5.2 Gefahren

Generell sind ADRRM-Systeme aufgrund einer Toleranzengleichheit zwischen System und Nutzern gegen schädliches Verhalten abgesichert. Die auf *Kollektiver Intelligenz* und evolutionärer Entwicklung basierende Systemstruktur weist allerdings einige Besonderheiten auf, die im Zuge der Betrachtung potentieller vom System ausgehender Gefahren besonders beachtet werden müssen.

5.2.1 Gefahren evolutionärer Anpassung

Gefahren aufgrund evolutionärer Anpassung bestehen in einer potentiellen Fehlentwicklung des Systems in Richtung nutzer- oder umgebungsgefähr-

dender Verhaltensweisen.

ADRRM-Systeme entwickeln sich evolutionär. Diesen Mechanismus benötigen sie zur Selbstanpassung an veränderte Umgebungsbedingungen und zur Implementierung neuer Systemelemente. Die evolutionäre Entwicklung vollzieht sich in ADRRM-Systemen über die Veränderungen der Verhaltensregeln. Dies geschieht über eine Modifikation der Schwellenwerte der Regelungsneuronen. Im Gegensatz zu lernfähigen Systeme entwickeln sich evolutionäre Systeme nicht gezielt in eine Richtung. Es besteht daher potentiell die Gefahr, dass sich evolutionäre Systeme in eine schädliche Richtung entwickeln. Um dem entgegenzuwirken, muss das System auf mehreren Ebenen abgesichert werden.

Die Absicherung auf der untersten Systemebene erfolgt über die bereits beschriebene Nutzer-/System-Toleranzengleicheiten. Diese stellt sicher, dass das System keine für den Nutzer gefährlichen Umgebungszustände herbeiführt.

Auf der nächsten Ebene verhindern unveränderliche Grenzwerte menschen- oder umgebungsgefährdende Aktionen. Diese Regeln sind in Form von Maximalwerten für die Regelneuronen festgeschrieben. Es handelt sich hierbei beispielsweise um festgeschriebene Anschläge für Servomotoren.

Auf einer höheren Programmebene muss die evolutionäre Entwicklung gedämpft werden. Das bedeutet, dass die Schwellenwerte der Regelneuronen vor zu großen Modifikationen geschützt werden. Große Veränderungen der Schwellenwerte können ansonsten zu sprunghaften und gefährlichen Verhaltensänderungen führen. Die evolutionäre Entwicklung verläuft aufgrund der Dämpfung langsamer und ist vor Überreaktionen geschützt. Temporär auftretende Umweltextrema führen nun nicht zu extremen Verhaltensänderungen.

Auf einer letzen Ebene wird auch ein langsamer Drift des Systems in extreme Verhaltensbereiche unterbunden. Hierzu erfolgt eine gelegentliche Einspeisung der ursprünglichen Startschwellenwerte. Anders ausgedrückt bedeutet dies eine partielle Rücksetzung, einen *Reset* der Schwellenwerte bei einem geringen Prozentsatz der Einheiten. Dieser Mechanismus führt das System immer wieder in Richtung der Startwerte zurück und verhindert so ein allzuweites Abdriften des Systems in extreme Richtungen.

5.2.2 Gefahr der Bildung neuronaler Netze

Neuronale Netze können emergent neue Fähigkeiten entwickeln, die eventuell nicht erwünscht sind. ADRRM-Systeme sind aus diesem Grunde nicht auf die Bildung *neuronaler Netze* angelegt. Aufgrund ihres evolutionären Entwicklungspotentials kann aber grundsätzlich nicht ausgeschlossen werden, dass sich in begrenzten Netzwerkbereichen dennoch funktionerende *neuronale Netze* (*neuronale Subnetze)* bilden.

ADRRM-Systemeinheiten können auch als künstliche Neuronen betrachtet werden. Sie sind in der Lage, diverse Eingangssignale über Schwellenwerte zu bestimmten Ausgangssignalen zu verarbeiten, die wiederum als Eingangssignale anderer Einheiten dienen. Dies entspricht der Arbeitsweise eines Neurons.
Durch die Verknüpfung mehrerer Neuronen entsteht ein *neuronales Netz*.
Die Fähigkeiten eines solchen Netzes ergeben sich aus der Sinnhaftigkeit der Verknüpfungen. Chaotisch verknüpfte Netze besitzen in der Regel keine Funktionalität, systematisch verknüpfte Netze können dagegen funktional sehr leistungsfähig sein. Die Wahrscheinlichkeit einer zufälligen Verknüpfung von Neuronen zu einem funktionsfähigen *neuronalen Netz* ist extrem gering, steigt allerdings mit zunehmender Systemgröße an. Hierbei ist zu bedenken, dass ein zwanzigstöckiges Bürogebäude bereits 6.000 bis 10.000 Systemeinheiten besitzen kann. Das einfachste *neuronale Netz* in der Fauna, mit Lernvermögen, besteht aus gerade einmal 302 Neuronen[1], die einfachsten als Gehirne bezeichenbaren *neuronalen Netze* bestehen aus nur 10.000 (*Drosophila melanogaster*[2]) bis 20.000 (*Aplysia ca-*

1 Vgl. TOSH, Colin; RUXTON, Graeme D.: Modelling Perception with Artificial Neural Networks. Cambridge 2010, S. 142.

2 Vgl. Scott et al. A Chemosensory Gene Family Encoding Candidate Gustatory and Olfactory Receptors in Drosophila. Cell, 104, New York 2001, S. 666.

lifornica[3]) Neuronen. *Neuronale Netze* mit einigen tausend Neuronen können demnach bereits Eigenschaften wie Lernvermögen oder auch Informationsspeicherung aufweisen. Im Falle großer ADRRM-Systeme mit einigen zehntausend Einheiten könnten sich daher zufällig durchaus *neuronale Subnetze* bilden. Diese Netze könnten dann beispielsweise in der Lage sein, dauerhaft Daten über Netzwerkverknüpfungsmustern zu speichern. Da ADRRM-Systeme nicht über eine dauerhafte Datenerfassung verfügen sollen, wäre eine zufällig auftretende Fähigkeit zur Langzeitspeicherung unerwünscht.

Um die Bildung funktionsfähiger neuronaler Netze zu unterbinden ist eine Begrenzung der Systemgröße und eine Abgrenzung einzelner ADRRM-Systeme voneinander notwendig.

5.2.3 Physikalische Unzerstörbarkeit

Umfangreichere dezentrale redundante Systeme sind physikalisch beinahe unzerstörbar.
Um ein dezentrales System vollständig abzuschalten, ist die einzelne Abschaltung eines großen Teils der Systemeinheiten notwendig. Ein solcher Abschaltvorgang erfordert bereits in kleineren Bürogebäuden die Abschaltung mehrerer tausend Einheiten. Würden sich die ADRRM-Systeme zu großen Clustern zusammenschalten, wäre eine komplette Abschaltung undurchführbar. Eine physikalische Abschaltung könnte nicht mehr erfolgen.
Eine Begrenzung der Systemgröße ist daher auch zur Wahrung einer physikalischen Systemabschaltung notwendig. Dies hat zur Folge, dass die theoretisch unbegrenzte Systemerweiterbarkeit aus Sicherheitsgründen nicht voll genutzt werden sollte.

3 Vgl. HERMEY, Guido et al.: Der Experimentator: Neurowissenschaften. Heidelberg, 2010, S. 4.

6 Anwendungsbeispiele

6.1 Anwendungsgebiete

ADRRM-Systeme sind in erster Linie für den Einsatz in öffentlichen und halböffentlichen Räumen bestimmt. In diesen oft unumgänglichen Räumen ist der Schutz der Privatsphäre besonderer erforderlich. Dies betrifft insbesondere Räume, in denen eine Standortbestimmung oder eine Wegverfolgung Rückschlüsse auf das Nutzerverhalten zulassen.

6.2 Gebäudeautomation

Das hier zur Prüfung der These entwickelte ADRRM System ist in erster Linie zur Gebäudeautomation von halböffentlichen Gebäuden mit relativ homogenen Nutzungen geeignet. Der Begriff der homogenen Nutzung bezieht sich auf einen konsistenten Zusammenhang zwischen Nutzeranforderungen und Nutzeraktivität. Dies bedeutet beispielsweise, dass der Nutzer eines Büros meist eine stabile Helligkeit und Temperatur für seinen Arbeitsplatz wünscht. Das System ist daher in der Lage, die Anwesenheit eines Nutzers mit invarianten Anforderungen an die Büroumgebung gleichzusetzen.
Eine realtiv invariante Regelung eignet sich für homogen genutzte Gebäuden. Beispiele hierfür sind: Bürogebäude, Schulen, Gastronomiebetriebe, Hochschulen, Institute, Museen und Ausstellungsgebäude, Bahnhöfe, Flughäfen, Hospitäler, Terminalgebäude, Sporthallen, Messehallen, Schwimmbäder, Hotelbetriebe, Werkstätten und Fabrikationsgebäude. All diese Gebäude können mit ADRRM-Systemen bei gleichzeitiger Wahrung des Nutzerdatenschutzes automatisiert werden. Eine Automatisierung kann in diesen Fällen sämtliche Bereiche umfassen, deren Automatisierungsanforderungen in Regeln fassbar sind. Grundvoraussetzung hierfür ist, dass sich die äußeren Bedingungen sensorisch erfassen lassen. Automatisierbar sind beispielsweise die Helligkeits- und Temperaturregelung, die bedarfsgerechte Steuerung von Aufzugsanlagen oder die Regelung der Wach- und Ruhephasen eines Gebäudes.
Die Vorteile der Automatisierung der oben genannten Nutzungsbeispiele durch ADRRM-Systeme gegenüber zentral gesteuerten, adressbasierten Systemen bestehen in der extremen Datensparsamkeit, der Nutzeranonymität, der unproblematischen Erweiterbarkeit, der Energieeffizienz des Steuerungssystems und der sehr hohen Betriebssicherheit.

Ein Einsatz in Wohngebäuden ist nicht ausgeschlossen, allerdings setzt die inhomogene Raumnutzung der Wohnräume ein verstärktes Eingreifen der Nutzer voraus. Wohnräume werden teilweise zu sehr unterschiedlichen Aktivitäten genutzt. Dementsprechend diffizil ist es für ein Automatisierungssystem zwischen den beabsichtigten Nutzungen zu differenzieren. In der Folge ist es dem ADRRM-System nicht möglich, sein Automatisierungspotential voll auszuschöpfen. ADRRM-Systeme werden daher nicht für den Einsatz in Wohngebäuden empfohlen.

6.3 Automation urbaner Räume

Urbane Umgebungen gehören zu den in Bezug auf den Nutzerdatenschutz besonders sensiblen Bereichen. Die Bevölkerung kann öffentliche Bereiche oft nicht umgehen oder ist auf die Nutzung dieser Bereiche angewiesen. Der daraus resultierende Nutzungszwang bedingt einen besonderen Schutz der Nutzer vor Positionsverfolgung oder anderweitiger Datenerfassung.
Die Nutzung urbaner Räume ist in der Regel homogen und somit ideal zur Automatisierung durch ADRRM-Systeme. In urbanen Räume sind ADRRM-Systeme insbesondere zur bedarfsgerechten Steuerung nutzerabhängiger Infrastrukturen geeignet. Dies betrifft unter anderem die energieeffiziente Steuerung von Wege- und Straßenbeleuchtung, die Aktivierung von Rolltreppen, den Betrieb von Brunnen und Wasserspielen und *die bedarfsgerechte Steuerung von Lichtzeichenanlagen, die bereits virtuell mit dezentraler Steuerung simuliert wurde*.[1] [2] ADRRM-Systeme regeln in den beschriebenen Anwendungsfällen die Infrastruktur bedarfsgerecht. Dies geschieht, indem das System erfasst, in welchen Bereichen sich Nut-

1 Vgl. LÄMMER, Stefan: Reglerentwurf zur dezentralen Online-Steuerung von Lichtsignalanlagen in Straßennetzwerken. Dissertation, Fakultät Verkehrswissenschaften "Friedrich List", Technischen Universität Dresden, Dresden 2007.

2 Vgl. GANDOLFI, Alberto: Menschen und Ameisen. Zürich 2001, S. 221.

zer aufhalten, in welche Richtung sie sich bewegen und wie weit sie von einem zu aktivierenden Teil der Infrastruktur entfernt sind (siehe Abbildung 114). Obwohl diese Vorgänge dezentral und anonym ablaufenn kann ein ADRRM-System die Infrastruktur inklusive einer entsprechenden Vor- und Nachlaufsteuerung regeln.[3] Der Vorteil von ADRRM-Systemen gegenüber herkömmlichen Steuerungssystemen besteht vor allem in der Wahrung der Nutzeranonymität in Kombination mit Ausfallsicherheit, Angriffsicherheit und unproblematischer Skalierbarkeit.

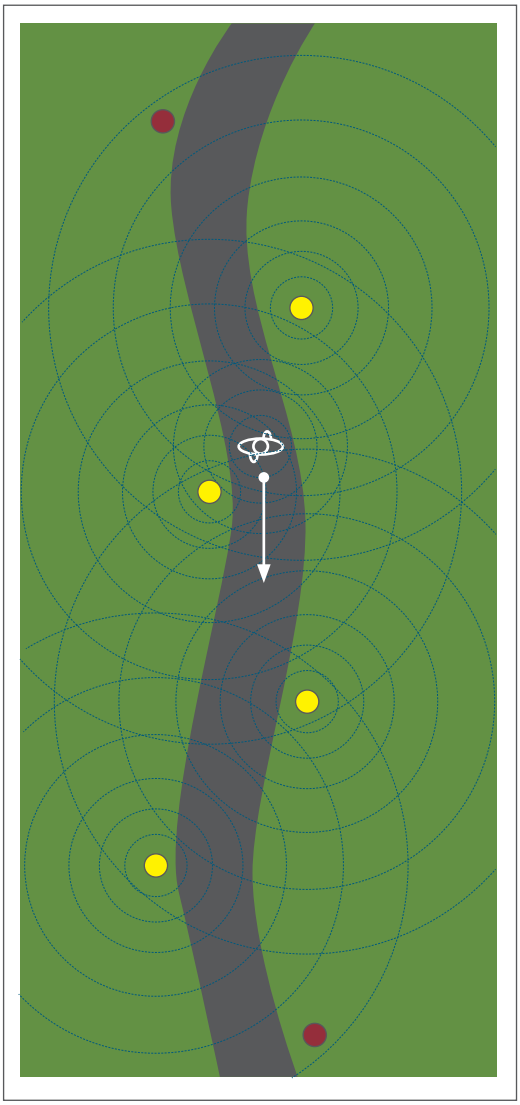


Abbildung 114:
Anwendungsbeispiel: Beleuchtungssteuerung

3 siehe auch die im Hotelmetapher-Metamodell beschriebenen Beispiele

6.4 Car-to-Car-Kommunikation

Im Gegensatz zu zentral gesteuerten Car-to-Car-Sicherheitssystemen, ist es mit ADRRM-Systemen möglich, eine Kommunikationsstruktur zur Wahrung der Verkehrssicherheit ohne Positionsbestimmung oder zentraler Überwachung der Kfz herzustellen.
Herkömmliche Systeme, die positionsbasiert und zentral arbeiten, können Positionsdaten unter anderem auch für lokal basierte Werbedienste, zur Ermittlung von Bewegungsprofilen und von Behördenseite aus, zur automatisierten Ordungswiedrigkeits- oder Strafverfolgung nutzen. Positionsbasierte Systeme ermöglichen beispielsweise die Erfassung jeglichen Verstoßes gegen die StVO im Bereich der Geschwindigkeitsübertretung oder des Falschparkens. Diese Systeme werden damit nicht dem Grundsatz der Datensparsamkeit gerecht, der im *BDSG* gefordert wird.[4] Wie das Beispiel des *TollCollect-Systems*, mit dem die deutsche LKW-Maut erhoben wird, zeigt, wecken Positionsdaten Begehrlichkeiten. Ein Zugriff zum Zwecke der Strafverfolgung wurde in einem Fall bereits erlaubt.[5] Eine Erstellung von Bewegungsprofilen und eine Speicherung derselben an einer zentral zugänglichen Stelle ist in ADRRM-Systemen nicht möglich und eine missbräuchliche Verwendung oder eine Verwendung zur Strafverfolgung ist damit ausgeschlossen.

Eine Kfz-Kommunikation zur Vermeidung von Unfällen, zur Warnung vor Stauenden oder zur Warnung vor Verkehrsteilnehmern im Toten Winkel kann mit Systemen nach dem ADRRM Prinzip anonym und ohne die zentrale Erfassung von Positionsdaten erfolgen. Die Ermittlung der Fahrzeugabstände erfolgt hierbei über die Reichweitenbegrenzung der Fahrzeugsystemeinheiten. Bewegt sich ein Fahrzeug in den Empfangsbereich eines anderen, wird über eine gestaffelte Sendeleistungsreduzierung der Abstand ermittelt. Verringert sich der Abstand unter einen festgelegten Wert, wird eine Geschwindigkeitsanpassung vorgenommen. Nähert sich ein weiteres Fahrzeug diesen beiden Kfz, wird es vorsorglich gewarnt.

4 Vgl. Bundesministerium der Justiz: Bundesdatenschutzgesetz §3a. (http://www.gesetze-im-internet.de/bdsg_1990/__3a.html), 26. April 2011.

5 Vgl. Verein zur Förderung des öffentlichen bewegten und unbewegten Datenverkehrs e.V.: Bürgerrechtsorganisationen: Straßen-Totalüberwachungs-Vertrag mit TollCollect kündigen. (www.foebud.org/misc/Stopp-TollCollect.pdf), 18. Mai 2011, S. 4-5.

Das Warnsignal kann auf eine bestimmte Anzahl von Weiterleitungsschritten begrenzt werden. Auf diese Weise ist es möglich, eine definierte Anzahl von Fahrzeugen vor einer Gefahrensituation zu warnen. Werden Richtantennen eingesetzt, ist auch die Ermittlung der Richtung, aus der ein Signal eintrifft, möglich. Dies ermöglicht zusätzlich eine Richtungsangabe zur Gefahrenquelle (siehe Abbildung 115).

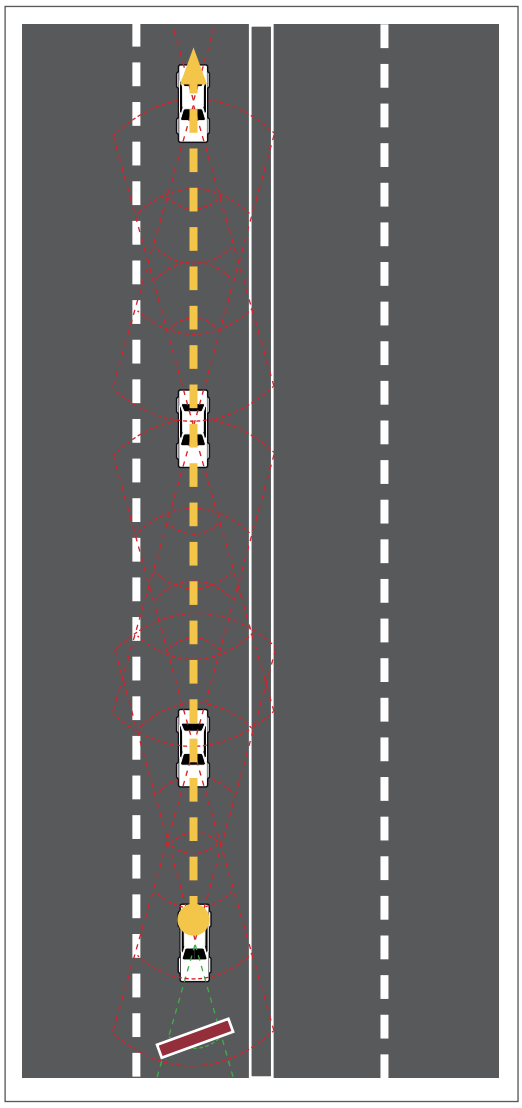


Abbildung 115:
Anwendungsbeispiel: Car-toCar-Kommunikation

Wird ein solches Car-to-Car-System mit einem ADRRM-System zur Lichzeichenanlagen-Steuerung kombiniert, sind weitere bedarfsgerechte Steuerungsvorgänge möglich. Rettungsfahrzeuge können sich in solchen Systemen den Weg regelrecht freischieben. Dies geschieht, indem sie den Ampelanlagen vor ihnen ihre Priorität anzeigen und diese eine Grüne Welle entlang der Fahrweges aufbauen.
Bei ADRRM Systeme kann auch auf das *Hovering-Cloud-Prinzip* zurückgegriffen werden.[6] Dieses Prinzip erlaubt auch dezentralen Systemen mit mobilen Einheiten eine ortsstabile Datenübertragung, indem diese Daten über mehrere Fahrspuren hinweg von einem Kfz zum anderen weitergegebn wird und so eine stehende Datenwolke, eine *Hovering-Cloud* generiert wird. Über dies Datenwolke können Kfz beispielsweise vor Gefahren wie Staus oder Unfällen gewarnt werden (siehe Abbildung 116)

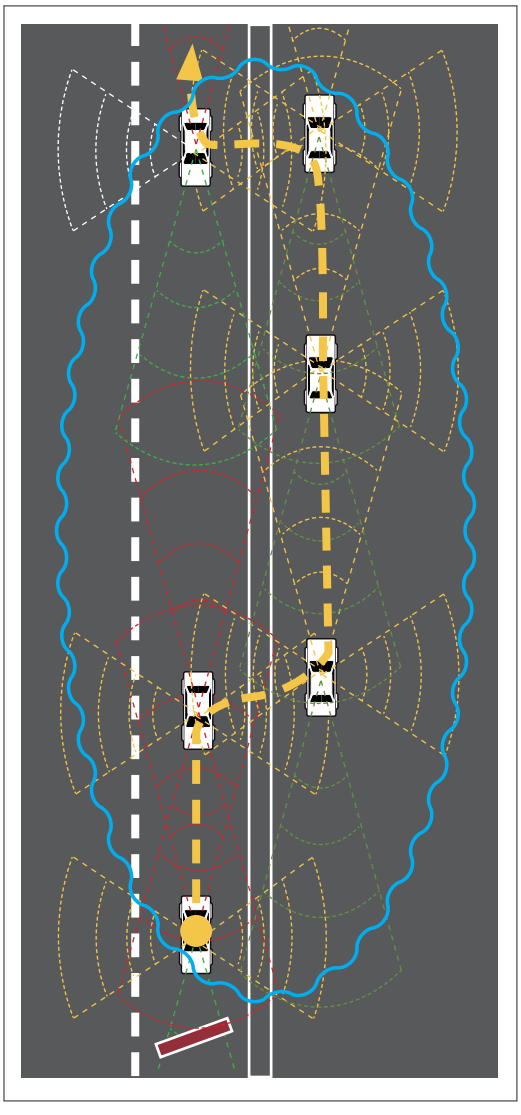


Abbildung 116:
Anwendungsbeispiel: Car-to-Car Hovering-Cloud

6 WEGENER, Axel; HELLBRÜCK, Horst et.al: Designing a Decentralized Traffic Information System AutoNomos, in: Kommunikation in Verteilten Systemen (KiVS), Hrsg. Klaus David, Kurt Geihs, Berlin 2010, S. 311

6.5 Medieninstallationen

Falls Medieninstallationen in einem größeren Umfang in halböffentlichen oder öffentlichen Räumen eingesetzt werden, werden auch diese zu potentiellen Einsatzgebieten für ADRRM-Systeme.
ADRRM-Systeme können die energieeffiziente Steuerung von Medieninstallationen übernehmen. Da diese Installationen zum Teil sehr viel Energie konsumieren, sollten sie nur aktiviert sein, wenn auch Betrachter anwesend sind.
Des Weiteren sind einige Medienfassaden interaktiv ausgelegt und erfassen zu diesem Zweck beispielsweise die Position von Passanten (siehe Abbildung 117).

Abbildung 117:
Anwendungsbeispiel: Medienkunstinstallation

Installationen, die einen umfangreicheren urbanen Bereich umfassen, zum Beispiel weite Bereiche von Fußgängerzonen, sollten ebenfalls mit datensparsamen und anonymen Systemen geregelt werden, da hier eine Positionsdatenerfassung in datenschutzrelevantem Ausmaß entstehen kann.

Messe- und Ausstellungsbereiche sind ebenfalls gut für den Einsatz von ADRRM-Systeme geeignet. Viele Medieninstallationen in Messe- und Ausstellungsräumen laufen in aktuellen Projekten im Dauerbetrieb und wirken sich störend auf den restlichen Ausstellungsbetrieb aus. Einfach einsetzbare dezentral gesteuerte Systeme eigenen sich gut zur interaktiven und bedarfsgerechten Steuerung von medialen Exponaten oder Videoinstallationen. Diese sind dann beispielsweise nur in Betrieb, wenn sich Personen vor den Installationen aufhalten und bleiben inaktiv, solange sich viele Besucher in nahegelegenen Bereichen der Ausstellung aufhalten, die durch die Installationen gestört würden. Wenn sich viele Besucher vor der Installation aufhalten und nur wenige in angrenzenden Bereichen, wird die Installation aktiviert. Ergänzend ist eine besucherabhängige Lautstärkenanpassung denkbar.

7 Resümee

These

Diese Arbeit verfolgt den Ansatz, das nutzerdatensparsamste und nutzerdatensicherste Steuerungs- und Automatisierungssystem für den Einsatz im Architekturbereich zu entwickeln. Dieser Ansatz wurde in den Anforderungen der These konkretisiert und niedergelegt:

Ein Automatisierungs- und Steuerungssystem zur Anwendung in Gebäuden und *urbanen Räumen*, das extrem datensparsam und nutzeranonym arbeitet und parallele sowie einfache lineare Prozesse steuern kann, selbstanpassend ausgelegt und individuell beeinflussbar ist, das den Schutz der Nutzer- und Systemdaten bereits in seiner Systemstruktur über eine vollständig dezentrale Systemorganisation ohne zentrale Zugriffs- und Angriffspunkte und adresslose Systemeinheiten gewährleistet, ist realisierbar.

Die These wird über ein vierstufiges Verfahren überprüft:

1) Metaphorisches Metamodell

In diesem Modell wird untersucht, ob eine Organisationsstruktur, welche die Anforderungen der These erfüllen kann, grundsätzlich vorstellbar ist. Dies wird über ein abstraktes Metamodell durchgeführt.
Das *Hotelmetaphermodell* liefert hierzu ein kohärentes Funktionsmodell. Dieses Metamodell erbringt den Nachweis, dass eine Organisation automatisierter Steuerungsvorgänge auch anhand anonymer bzw. nicht adressierbarer Einheiten möglich ist. Dies erfordert allerdings eine physikalische Begrenzung der Informationsausbreitung. Diese wird in der Hotelmetapher über die Reichweiten der Stimmen der Gäste und Angestellten umgesetzt. Des Weiteren wird nachgewiesen, dass eine individuelle Umgebungsbeeinflussung in verschiedenen räumlichen Größenordungen möglich ist. Eine Erweiterung des Aktivierungsbereiches über den Radius der eigenen Stimme hinaus ist durch eine schrittweise Weiterleitung der Gästewünsche von einem Angestellten zum nächsten möglich. Im Falle einer Gefahrensituation wird ebenso verfahren, allerdings ohne Begrenzung der Weiterleitungsschritte. Ebenso wird nachgewiesen, dass dieses System selbstanpassend ausgelegt werden kann. Dies kann gleichfalls dezentral geschehen, indem Hotelangestellte, die suboptimale Verhaltensweisen aufweisen, durch neue Mitarbeiter ersetzt werden, die leicht variierte Verhaltensweisen besitzen und die ihren Aufgaben eventuell besser nachkommen können.

2) Natürliche Metamodelle

In einem nächsten Schritt wird nach existierenden Systemen gesucht, die ebenfalls aus Kollektiven mit anonymem Einheiten bestehen. Diese Systeme finden sich in den gut untersuchten Kollektiven staatenbildender Insekten.
Die pure Existenz dieser Systeme beweist die grundsätzliche Funktionsfähigkeit von Kollektiven mit dezentralem Aufbau und anonymen Mitgliedern. Die Leistungsfähigkeit dieser natürlichen Systeme beweist zudem, dass Kollektive mit dezentralem Aufbau und anonymen Mitgliedern in der Lage sind, sehr umfangreichen und komplexen Aufgaben nachzukommen. Die grundsätzliche Eignung dezentraler Systeme mit anonymen Einheiten zur Automatisierung umfangreicher Umgebungen ist damit bewiesen.
Zudem zeigen Studien über die Verhaltensweisen staatenbildender Insekten einen Teil der Organisationsprinzipien auf, die in ein künstliches System übernommen werden können.

3) Theoretisches künstliches System

Die Metamodelle weisen nach, dass die Anforderungen der These durch dezentrale Systeme mit anonymen Einheiten erfüllt werden können. Die Metamodelle liefern zudem auch einen Großteil der benötigten Organisations- und Kommuniaktionsprinzipien.
Die in den Metamodellen gefundenen Prinzipien und Strategien werden nun in einem theoretischen künstlichen System zusammengeführt. Es stellt sich heraus, dass es bereits künstliche dezentrale Systeme mit anonymen Einheiten gibt. Diese werden nach Quellenlage allerdings noch nicht zur Automatisierung im Architekturbereich verwendet. Der

Forschungsstand auf dem Gebiet der künstlichen Kollektive ermöglicht eine systemtheoretische Einordnung des Steuerungssystems in verschiedene systemtheoretische Klassen. Es handelt sich um ein *Multi-Agenten-System*, das *dynamische nichtlineare Verhaltensweisen* zeigt, dessen Agenten teilweise *Fuzzy-Logik* basiert sind und das sich den *Komplexen-Adaptiven-Systemen* zuordnen lässt.

Das theoretische künstliche System wird ein A*dressloses-Redundanzbasiertes-Reichweitebeschränktes-Minimalinformationssystem* werden: ein ADRRM-System.

Das theoretische ADRRM-System besteht aus einem dezentral organisierten Kollektiv autonomer Einheiten. Diese Einheiten basieren auf Verhaltensregeln. Diese Regeln sind in Form von Regelkreisen angelegt. Die Regelkreise bestehen aus Potentialen, die sich durch den Einfluss multipler Faktoren heben oder senken. Ihre Reaktionen erfolgen auf Über- oder Unterschreitung von Schwellenwerten. In ADRRM-Systemen können auch zwei Schwellenwerte pro Regelkreis angelegt sein. Ist dies der Fall, handelt es sich um einen Regelkreis mit einer Art von *Fuzzy-Logik*. Regelkreise, die multiple Einflüsse zu einem Ausgangssignal verarbeiten können, ähneln natürlichen Neuronen. Sie werden daher in dieser Arbeit auch als künstliche Neuronen betrachtet. Die Möglichkeit eines *neuronalen Netzes* wird in dieser ADRRM-Systemstruktur aus technischen Gründen nicht wahrgenommen. Umfangreichere ADRRM-Systeme könnten jedoch unbeabsichtigt *neuronale Subnetze* bilden, wie unter dem Punkt *Gefahren* geschildert.

ADRRM-Systeme bestehen aus einer großen Zahl von Einheiten mit komplexen Abhängigkeiten und *Fuzzy-Logik-Verhaltensweisen*. Die so entstehenden Kollektive werden sich *deterministisch chaotisc*h verhalten. Ihr Verhalten wird dem Beobachter daher *komplex* und *emergent* erscheinen. Um das Verhalten der Einheiten über den Tag zu strukturieren und einer zu starken *Emergenz* vorzubeugen, werden die Verhaltensweisen der Einheiten über *Beriebsmodi* geregelt. Diese *Modi* richten sich nach Umgebungsereignissen und legen die Metaverhaltensweisen fest. Das bedeutet, die *Modi* legen fest, wie reaktionsfreudig sich die Einheiten verhalten und regeln bestimmte Systemvorgänge wie beispielsweise den evolutionären Prozess. Die *Modi* sind zwar in der Regel nicht systemweit synchronisiert, werden aber aufgrund der geringen Schwellenwertvarianz zwischen den einzelnen Einheiten weitgehend parallel auftreten. Somit werden sich die *Modi* in der Regel auch in Form *globaler Modi* ausprägen.

ADRRM-Systeme sind grundsätzlich *ereignisbasiert* angelegt. Diese Strategie wurde gewählt, um den Systemen ein flexibles Verhalten zu ermöglichen. Eine manuelle Veränderung einer zeitbasierten Verhaltensregelung wäre in dezentralen Systemen eine sehr aufwendige Angelegenheit und wird daher in ADRRM-Systemen nicht in Betracht gezogen.

Da sich die Umgebungsbedingungen dermaßen umfangreich verändern können, dass eine Anpassung des Systemverhaltens notwendig ist, müssen ADRRM-Systeme adaptiv ausgelegt werden. Da auch dies manuell in dezentralen Systemen beinahe unmöglich ist, ist eine Selbstanpassung unumgänglich. Eine Anpassung über *Lernverhalten* würde vorgegebene Ziele erfordern. Diese würden die Flexibilität der Entwicklung stark einschränken. Zudem müsste das System über eine sehr feine Sensorik verfügen, um die Auswirkungen der eigenen Verhaltensweisen von denen anderer zu unterscheiden. Des Weiteren wäre eine Langzeitspeicherung diverser Umgebungs- und Verhaltensdaten nötig, um eine Beurteilungsbasis für den Lernerfolg zu ermöglichen. Aus diesen Gründen wurde in ADRRM-Systemen von einer Anpassung über Lernverhalten Abstand genommen.
Stattdessen wird eine *evolutionäre Anpassung* vorgenommen. Diese ist mit wesentlich geringeren technischen Ressourcen umsetzbar und gibt keine Entwicklungsrichtung vor. Die Beurteilung der Verhaltensweisen erfolgt rein über die Überlebensfähigkeit der einzelnen Einheit. Die Überlebensfähigkeit hängt von der Fähigkeit der Einheit ab, ihre Umgebung so zu regeln, dass diese ihr Bedingungen bietet, die innerhalb der Überlebenstoleranzen der Einheit bleiben. Bewegen sich die Umgebungsbedingungen in Bereiche außerhalb der Überlebenstoleranzen der Einheit, wird sie *krank*. *Kranke* Einheiten erhalten in dem nächsten *Evolutionsmodus* neue Schwellenwerte von den verbleibenden *gesunden* Einheiten.

Sie verrechnen diese mit ihren bisherigen Schwellenwerten und ersetzen diese durch die neu berechneten. Die ehemals *kranken* Einheiten werden so zu der nächsten Generation. Dieser Prozess beweist, dass eine Selbstanpassung in dezentralen Systemen mit einfach aufgebauten Einheiten möglich ist.

Von der evolutionären Entwicklung ausgenommen sind die Verhaltensweisen zur *Gefahrenabwehr*. Diese könnten sich zwar theoretisch ebenfalls evolutionär ausprägen. Dies würde aber die Freigabe mehrerer Gebäude oder anderer zu steuernder Umgebungen erfordern. Eventuell ließe sich dies in virtuellen Umgebungen umsetzen. Die erfolgreichen *Gefahrenabwehrverhaltenweisen* würden dann in die Systeme übernommen, die reale Umgebungen steuern. Der wahrscheinlichere Fall besteht allerdings in der manuellen Voreinstellung der *Gefahrenabwehrverhalten*.

Die möglichen Einsatzbereiche der ADRRM-Systeme sind einzig durch drei Faktoren beschränkt:
Erstens müssen sich die global erwünschten Verhaltenweisen in auf die Einheiten der Kollektivs *verteilbare lokale Regeln* herunterbrechen lassen.
Zweitens legen die erreichbaren *Reichweitenradien* die räumliche Auflösung der ADRRM-Systeme fest. Liegt die geringste Auflösung bei einem Radius von einem Meter, kann das System keine Räume unter diesem Radius differenziert steuern. Im umgekehrten Fall, der im urbanen Bereich auftreten kann, wirkt sich die größtmögliche erreichbare Reichweite auf die maximale räumliche Systemauflösung nicht einschränkend aus; die maximale Reichweite kann durch Einfügen weiterer Einheiten vervielfacht werden. ADRRM-Systeme sind somit in ihrer geringsten räumlichen Steuerungsauflösung technisch eingeschränkt, in ihrer weitesten aber unbeschränkt. Ein Einsatz in *urbanen Räumen* ist daher gut möglich.
Drittens sind die Steuerungsmöglichkeiten der *ADRRM-Systeme* auf eine Vielzahl einfacher *paralleler Tätigkeiten oder auf einfache lineare* Tätigkeiten beschränkt. Zur zeitkritischen Steuerung umfangreicher linearer Prozesse sind sie nicht geeignet.

Neben dem radikalen Schutz der Nutzerdaten bieten Systeme nach dem *A*DRRM-Prinzip einige weitere Vorteile. Aufgrund ihrer dezentralen Organisationsstruktur, ihrer redundanten Hardwareeinheiten und der r*edundanzbasierten Kommunikationsstrukt*ur sind ADRRM-Systeme extrem ausfallsicher und auch physisch nahezu unzerstörbar. Des Weiteren wird der Teil den Infrastruktur, der bei zentralen Systemen für die Steuerungsserver benötigt wird, obsolet. ADRRM-Systeme benötigen keine Steuerungsmodelle, ADRRM-Systeme sind daher universeller einsetzbar als zentral gesteuerte Systeme. Da ADRRM-Systeme direkt in der gesteuerten Umgebung arbeiten, ihre Einheiten in Echtzeit reagieren und die Dichte der Einheiten ohne Rücksicht auf Steuerungsmodelle exakt an die Gegebenheiten angepasst werden kann, können ADRMM-Systeme die Umgebung detaillierter erfassen und steuern. Dies sollte zu einer energieeffizienteren Regelung der Umgebung führen.

Von ADRRM-Systemen können gleichwohl auch konzeptbedingte Gefahren ausgehen. Einige Vorteile des ADRRM-Systems können sich je nach Standpunkt des Betrachters auch als Gefahren erweisen. So ist die physische Unzerstörbarkeit ein Vorteil in Bezug auf die Betriebssicherheit, sie kann aber auch als Nachteil in Bezug auf die Kontrollierbarkeit des System betrachtet werden. Eine physische Abschaltung von ADRRM-Systemen kann einzig über eine manuelle Abschaltung eines Großteils der Einheiten erfolgen. Dies ist in umfangreichen Systemen beinahe unmöglich. Aus diesem Grunde muss der Schutz der Nutzer und der gesteuerten Umgebung auf der untersten Systemebene gesichert werden. Dies bedeutet, dass das System schon aus eigenem Interesse nicht in der Lage sein darf, nutzer- oder umgebungsschädigende Verhaltensweisen durchzuführen. Dies wird in ADRRM-Systemen über die Überlebenstoleranzwerte der Einheiten geregelt. Diese sind interessengleich mit den Nutzermindestanforderungen angelegt. Steuert eine Einheit die Umgebung aus diesem Bereich heraus, eliminiert sie sich somit selbst.
Eine weitere wesentlich abstraktere Gefahr besteht in der Bildung *neuronaler Subnetze*. Dezentrale Netze mit neuronenähnlichen evolutionär veränderbaren Einheiten sind grundsätzlich in der Lage, zufällig *neuronale Netze* zu bilden. Dies ist nicht erwünscht, da *neuronale Netze* aufgrund ihrer *emergenten* Fähigkeiten Verhaltensweisen wie beispielsweise verbindungsbasierte Langzeitdatenspeicherung ausprägen können. Die so entstandenen Fähigkeiten

können konzeptionell unerwünscht oder gar gefährlich ausfallen. Um dies zu verhindern, müssen einzelne ADRRM-Systeme voneinander separiert werden, um die globalen Systemgrößen auf ungefährliche Ausmaße zu begrenzen. Wo diese Grenzen liegen, konnte anhand der durchgeführten Versuche nicht geklärt werden.

4) Reales Versuchssystem

Nach dem Nachweis, dass sich die Anforderungen der These durch ein System nach dem ADRRM-Prinzip theoretisch erfüllen lassen, wurde ein realer Versuch durchgeführt. In einem *proof-of-concept-Versuch* wurde die reale Umsetzbarkeit und Funktionalität der grundlegenden Prinzipien überprüft.
Der Versuch bestand aus einem Steuerungskollektiv aus bis zu vierzehn Einheiten, das in einem bestehende Mauerwerksrohbau montiert wurde. Die sechs Steuerungseinheiten waren in der Lage, Aktuatoren in Form von Fassadenklappen zu bewegen. Diese Klappen wurden pro gesteuertem Wandmodul in zwei Gruppen zusammengefasst, die untere Klappengruppe diente der Belüftung, die obere Gruppe der Verschattung. Bedauerlicherweise war es aufgrund der vorgegebenen baulichen Situation nicht möglich, das Innenraumklima durch die Klappen signifikant zu beeinflussen. Damit war ein Nachweis einer evolutionären Entwicklung in eine sinnvolle Richtung nicht möglich. Des Weiteren wurden zwei außenliegende Einheiten als Sensoreinheiten angebracht. Zwei weitere Einheiten wurden zur Detektion von Personen verwendet, eine im Außenbereich zur Eindringlingsgefahrenerkennung und eine im Innenraum zur Nutzerdetektion. Die Signalreichweiten der Steuerungs-, Sensor- und Personendetektionseinheiten überlappten sich zunächst nicht in allen Bereichen. Daher wurden zwei weitere Einheiten als Funkbrücken eingesetzt. Zur Überprüfung des individuellen bereichsbeschränkten Nutzereinflusses wurde eine Mobileinheit konstruiert, über die ein Nutzer zu einem Teil des Kollektivs wurde und somit in der Lage war, *Wünsche* an das System zu übermittel. *Wünsche* trifft es insofern, als dass ein Nutzer in ADRRM-Systemen nur *Einfluss,* aber nie *Befehlsgewalt* besitzt.
Die Programmierung der Einheiten umfasste die *Adresslose Kommunikation* mit *selbst anpassender physikalischer Signalreichweitenbeschränkung*, *Wach- und Ruhemodi*, einen vereinfachten *Appellmodus* mit anschließenden *Evolutionsmodus*, einen *Eindringlingsgefahrenmodus* und einen *Windgefahrenmodus*. Das System arbeitete entsprechend den ADRRM-Prinzipien *absolut nutzeranonym* und *extrem datensparsam*.

Die Versuchsläufe zeigten, dass die theoretisch entwickelte Strategie der Kommunikation über allgemein verwendete *Minimalinformationen* ohne spezifische Sensorwerte und *ohne Adressierung* funktionstüchtig war. Ebenso funktionierte die *individuelle bereichsbeschränkte Umgebungsbeeinflussung*. Auch die *globale Weiterleitung* konnte am Beispiel der Gefahrensignale erfolgreich nachgewiesen werden. Die Einheiten waren zudem in der Lage, ihren Signalreichweitenradius an veränderliche Umgebungsbedingungen anzupassen. Das Versuchssystem steuerte parallel achtundvierzig Klappen, die in insgesamt zehn Gruppen zusammengefasst waren und wertete dazu die Informationen von bis zu achtzehn Sensoren und eine Vielzahl von Funksignalen aus. Der Nachweis der *Fähigkeit zur parallelen Prozessbearbeitung* war damit ebenfalls erbracht.
Zwei Punkte der These konnten in dem Versuchsaufbau nicht direkt überprüft werden. Zum einen gab es keine linearen Prozesse, die hätten gesteuert werden können und der Versuch erreichte nicht die Ausmaße *urbaner Räume*. Die Fähigkeit zur Steuerung einfacher linearer Aufgaben bleibt damit theoretisch. Der Nachweis der Einsatzfähigkeit in Dimensionen *urbaner Räume* konnte einzig indirekt erfolgen: Da das System räumlich in verschiedenen Abstufungen arbeiten konnte und die räumliche Ausdehnung über eine simple räumliche Addition weiterer Einheiten erfolgen kann, wäre es möglich, das funktionierende System mehrfach zu addieren und so einen Raum beliebiger Größe, auch in urbanen Dimenionen, abzudecken.

Abschließend lässt sich belastbar behaupten, dass die These anhand des theoretischen vollständig und des realen ADRRM-Systems in weiten Teilen als zutreffend bestätigt werden konnte. ■

8 Programmierung

Die Programmierung stammt, soweit nicht anders erwähnt, vom Autor.

Programmiert wurde in *Arduino* mit der *Arduino-Entwicklungsumgebung* unter Zuhilfenahme der jeweils unter „Bibliotheken“ angegebenen Programmbibliotheken diverser Autoren. Die Dokumentation dieser Bibliotheken, ihrer Autoren und der *Arduino-Entwicklungsumgebung* findet sich im Internet unter:

http://www.arduino.cc/playground/

ADRRM-Progammierung der Steuereinheiten Beta Version 0.97; mit korrigiertem Evolutionsmodus

mit bereinigten Kommentaren

Beschreibung

```
// ADRRM Beta V0.97 - 7.Dez. 2010 von Martin Kim
// copyright 2011 by Martin Kim

// Funksignale im System
// 99-Gefahr (Eindringling), 88-Gefahr Wind, 77-Sensoren außen,
   33 Person innen, 66-Evolution, 55-Apell, 44-Mobilmodul anwesend
   30 zu kalt, 31, zu warm, 21 zu hell, 20 zu dunkel, 11-PING
```

Verwendete Ardunio-Bibliotheken
Servosteuerung: Servo.h
Transceiversteuerung: Nrf2401.h

```
// Bibliotheken
#include <Servo.h>          // Servo Bibliothek
#include <Nrf2401.h>        // Transceiver/Funk Bibliothek
Nrf2401 Radio;
```

Variablen - global

```
// Globale Variablen
// Sonstige
unsigned long millis();              // Zeit seit letztem Reset
unsigned long lebenszeit = 0;        // Lebenszeit
long zustandsWert = 10;              // Startwert Gesundheit

// Wert auf Basis von Funksignalen
der aufgerechnet wird, um eine bestimmte Aktion auszulösen
int mobHell = 0;                     // für Helligkeit
int mobTemp = 0;                     // für Temperatur

// hier wird hinterlegt, was gesendet wurde, um einen Rundlauf der Funksignale
zu vermeiden
int gesendet = 0;
int serialFlag = 9999;               // Zeitspanne für Serielle Sendepause

// Led blinkt im Wachzustand, wenn Modul „krank“ wurde, ansonsten keine
Pause = keinBlinken
int ledpause = 0;
```

Variablen der Systemmodi

```
// Modi
// Wach Ruhe Startwerte der Schwellenwerte
int swMHo = 960;                     // hell
int swMHu = 1010;                    // dunkel
int swMTo = 90;                      // warm
int swMTu = 2;                       // kalt
long WRpot = 10000;                  // Wach-/Ruhe Potential
int swWRu = -5000;                   // Schwelle Wach-/Ruhe unten
int swWRo = 5000;                    // Schwelle Wach-/Ruhe oben
int flagWR = 1;                      // Speichert den Systemzustand ab,
                                        Ruhe = 0, Wach = 1
int flagWRalt = 1;                   // alter Flagwert
```

noch Variablen der Systemmodi

```
int modWRwert = 1;          // Flag f ü r Wach Ruhe im Neuron
int nWert = 1;              // Flag für Wach Ruhe in neuronPot
int harmH1 = 1;             // harmonisierter Rückgabewert
int harmH2 = 1;             // harmonisierter Rückgabewert
int harmH3 = 1;             // harmonisierter Rückgabewert
int harmT1 = 1;             // harmonisierter Rückgabewert
int harmT2 = 1;             // harmonisierter Rückgabewert
int harmT3 = 1;             // harmonisierter Rückgabewert
int sensorH = 0;            // Messwert Hell
int sensorT = 0;            // Messwert Temp
unsigned long restzeit = 0;// nicht mehr verwendet

// Gesundheit
int gesundheit = 1;         // Gesundheitszustandswert auf gesund setzen
// Schwellenwert max mit Zufallsvarianz damit leicht unterschiedliche Werte im
System sind
int swGHu = 940 + random (-5, 5);   // Schwellenwert max Dunkelheit
int swGHo = 100 + random (-5, 5);   // Schwellenwert max Helligkeit
int swGTu = 1 + random (-1, 1);     // Schwellenwert min Temp
int swGTo = 100 + random (-5, 5);   // Schwellenwert max Temp
long gesundPot = 1000;              // Startwert Gesundheit
int sigG = 0;                       // Einfluss Modulen außen / Mobileinheit
int sigG2 = 0;                      // Einfluss Modulen außen / Mobileinheit

// Evo
unsigned long evozeit = 0;          // Zeitraum Evolution

// Servos
Servo myservo1;                     // Servodefinieren für Bibliothek
Servo myservo2;                     // Servodefinieren für Bibliothek
int vz=1;                           // Vorzeichen für Servodrehrichtung
int thispos= 120;                   // Ausgangsposition
int thispos2=55;                    // Ausgangsposition
                                    (zupos 55 bei normal, 130 bei Modul7)
int endpos = 45;                    // Endposition mit Initialwert
int endpos2 = 45;                   // Endposition mit Initialwert
int aktion = 0;                     // ziehtspäter einmal Energie ab
int servoPot = 0;                   // zeitliche Dämpfung nach Servoauslösung
int servoPotTemp = 0;
int servowait = 0;                  // kann Servo länger in Position halten - z.B.
bei Alarm

// Radio
int sig00 = 0;                      // Signaltyp
int sig01 = 0;                      // Schwellenwerte im Evolutionsmodus
int sig02 = 0;
int sig03 = 0;
int sig04 = 0;
```

noch Variablen Systemmodi

```
int sig05 = 0;
int sig06 = 0;
/ / Variablen zur Seriellen Ausgabe für Loggs
int funk77 = 0;
int funk55 = 0;
int funk66 = 0;
int funk21 = 0;
int funk31 = 0;
int funk20 = 0;
int funk33 = 0;
int funk99 = 0;
int funk88 = 0;
int funk44 = 0;
int funk11 = 0;
int funk12 = 0;
int print155 = 0;
int print166 = 0;
int funk111 = 0;                // selber 11 gesendet (nicht empfangen)
int funk112 = 0;                // selber 12 gesendet (nicht empfangen)

int pingzeit = 0;               // nicht mehr verwendet ?
int sl = 1;                     // Sendeleistung
int rwflag = 0;                 // 0 = Reichweitenanpassung läuft nicht
int pingalt = 0;                // Reichweitenanp.vorhergehendes Ping
int pingneu = 0;                // Reichweitenanp. neues Ping
int zufall = 0;
```

Setup - Voreinstellungen

```
void setup () {
// Beginne mit Serieller Kommunikation zum loggen
Serial .begin (9600);

// maximale Lebenszeit: 2 Tage in Millisekunden + Startzeit zum Zeitpunkt des
Progammstarts/Reset
lebenszeit = 17280000 + millis ();

// Transceiver/Funk initialisieren
Radio.power = sl;               // Sendeleitung am Anfang auf maximum = 3
Radio.localAddress = 1;         // eigene Adresse - alle identische Adressen!!
Radio.remoteAddress = 1;        // von Sender - alle identische Adressen!!
Radio.rxMode(7);                // 7 byte zu empfangen

// Servos
myservo1.attach (10);           // Servo #1 ist an Analog Pin10
myservo2.attach (11);           // Servo #1 ist an Analog Pin11
myservo1.write (thispos);       // Servo #1 in Starposition fahren
myservo2.write (thispos2);      // Servo #2 in Starposition fahren
```

noch Setup

```
// Led
pinMode (13, OUTPUT);              // Modul Wach(Led an) oder Ruhe(Led aus)
digitalWrite (13, LOW);            // an Anfang erstmal ausschalten

// Zufallskennung Startwert setzen
// Zufallszahl auf Sig01 damit bei Reichweitenermittlung zwischen der Antwort
mehrerer Module unterschieden werden kann: Wert 0 ist raus - siehe 12
zufall = random (1, 999);

// Für asnychronen Start der Einheiten, damit es weniger Ueberlagerungen in
der Kommunikation gibt
int desynch= random (0, 500);
delay (desynch);

}                                  // Ende Setup
```

Ablaufkontrollebene - Loop

```
void loop () {
// zum Start Funk auf Empfang
Radio.rxMode(7);
```

Sprung zu Hauptroutine für den Fall dass keine Signale empfangen wurden

```
// ++++ HAUPTroutine ohne Empfang von Funksignalen ————————————————
while (!Radio. available ()) {
leben ();                          // Funktion Lebenszeit aufrufen
haupt1 ();                         // Hauptfunktion ohne Empfang aufrufen
}
```

Sprung zu Hauptroutine für den Fall dass Signale empfangen wurden

```
// ++++ HAUPTroutine bei Empfang von Funksignalen ————————————————
leben ();                          // Funktion Lebenszeit aufrufen
Radio.read ();                     // Daten aus Empfangspuffer auslesen
sig00 = Radio.data[0];             // Art der Nachricht - Signaltyp
sig01 = Radio.data[1];             // zu uebertragende Werte
sig02 = Radio.data[2];
sig03 = Radio.data[3];
sig04 = Radio.data[4];
sig05 = Radio.data[5];
sig06 = Radio.data[6];
// Hauptfunktion mit Empfang aufrufen (sigxx sind GLOBAL und müssen daher
nicht extra übergeben werden)
haupt2 ();

} // Ende loop
```

Hauptfunktionen

```
// Hauptfunktionen
```

Haupt 1 - ohne Singalempfang

```
// Haupt 1
void haupt1 () {
// Serial Print Timer - damit nicht permanent Daten gesendet werden, ansonsten
wird die Datenmenge zu umfangreich
serialFlag ++;
if (serialFlag > 10000) {       // Wenn Timer abgelaufen, dann neu starten
serialFlag = 0;
}

// Aufgewacht ?
// Aufgewacht bedeutet Wechsel von Ruhe=0 zu Wach=1
if (flagWRalt < flagWR) {
apell ();
}
else {
modWach(0,0);                   // keine Signale von außen: nichts zu übergeben - 0,0
}
}                               // Ende haupt1
```

Haupt 2 - mit Singalempfang

```
// Haupt 2
void haupt2 () {
// Serial Print Timer
serialFlag ++;
if (serialFlag > 10000) {       // Wenn Timer abgelaufen, dann neu starten
serialFlag = 0;
}

// Gefahr Eindringling
if (sig00 == 99) {
mobHell = -1000;                // Hell.-Potential 1000 abziehen = sofortige Aktion
mobTemp = -200;                 // Temp.-Potential 200 abziehen = sofortige Aktion
servoPot = servoPot - 8500; // überdrücke Dämpfung Servos = sofortige Aktion
servoPotTemp = servoPotTemp - 8500;
servowait = 10000;                              // Klappen bleiben länger zu
// Funk weiterleiten, letzte Zahl Anzahl der Wiederholungen
funk (99,99,99,99,99,99,99, 3);
//Notfall maximale Sendeleitung - fehlt hier, Fehler!
gesendet = 99;
funk99 = 99;                                    / / Variable serielle Ausgabe
}

// Gefahr Wind
if (sig00 == 88) {
mobHell = -1000;
mobTemp = -200;
```

noch Haupt 2

```
servoPot = servoPot - 8500;
servoPotTemp = servoPotTemp - 8500;
servowait = 100000; Radio.power = 3; //Notfall maximale Sendeleitung
funk (88,88,88,88,88,88,88, 3);
gesendet = 88;
funk88 = 88;
}

// Geweckt worden ? - dann Evolutionszeit starten
if (sig00 == 55) {
// Funk weiterleiten
funk (55,55,55,55,55,55,55, 5);

// Rundlauf zu vermeiden
// die 66 wird nicht weitergeleitet, daher keine Risiko eine Rundlaufs, die 55
könnte aber immer noch unterwegs sein
gesendet = 55;
funk55 = 55;

// Warte ein wenig, damit alle Zeit zum Aufwachen haben
delay (200);

//Evozeitraum begrenzen
// Evozeit max 10 Sekunden nach Start
evozeit = millis () + 10000;
// Falls gesund, dann sende deine Schwellenwerte
if (gesundheit == 1) {

// Werte zum Senden leicht mutieren
int swMHoSend = swMHo + random (-2, 2) ; // hell
int swMHuSend = swMHu + random (-2, 2);  // dunkel
int swMToSend = swMTo + random (-2, 2);  // warm
int swMTuSend = swMTu + random (-2, 2);  // kalt
int swWRuSend = swWRu + random (-5, 5);  // Schwelle Wach Ruhe unten
int swWRoSend = swWRo + random (-5, 5);  // Schwelle Wach Ruhe oben

//sende mutierte Schwellenwerte 10mal
funk (66,swMHoSend,swMHuSend,swMToSend,swMTuSend,swWRuSend,swW
RoSend, 10);
print166 = 166;                          // selber 66 gesendet
}                                        // Ende gesundheit

// fertig - WR Pot deutlich hoch setzen, damit sie aufwachen - weitermachen mit
WR Modus
modWach (60000,60000);
} // Ende 55
```

Evolutionsmodus korrigiert: Werte zur Übertragung runterrechnen auf ein byte

```
int swMHoSend = (swMHo + random (-20,
21)) / 10;

 int swMHuSend = (swMHu + random (-20,
21)) / 10;

int swMToSend = swMTo + random(-5, 6);

int swMTuSend = swMTu + random(-5, 6);

int swWRuSend = ((swWRu + random(-20,
20)) / 100) * -1;

int swWRoSend = (swWRo + random(-20,
20)) / 100; // **
```

noch Haupt 2

```
// Evolution !!!
if (sig00 == 66) {
funk66 = 66;

// Evolutionszeit temporär begrenzt - falls noch in der Zeit...
if (millis () <= evozeit) {

// Falls krank
if (gesundheit == 0) {
// Wach Ruhe Modus Werte modifizieren
// Alte Werte abspeichern
int swMHoAlt = swMHo;                    // hell
int swMHuAlt = swMHu;                    // dunkel
int swMToAlt = swMTo;                    // warm
int swMTuAlt = swMTu;                    // kalt
int swWRuAlt = swWRu;                    // WR Schwelle unten
int swWRoAlt = swWRo;                    // WR Schwelle oben
```

Evolutionsmodus korrigiert:
zur Übertragung runtergerechnete Werte werden wieder hergestellt

```
// neue Werte von gesunden/Spendermodulen speichern
int NswMHo = sig01 * 10;                 // hell
int NswMHu = sig02 * 10;                 // dunkel
 int NswMTo = sig03;                     // warm
int NswMTu = sig04;                      // kalt
int NswWRu = (sig05 * 100) * -1;         // WR Schwelle unten
int NswWRo = sig06 * 100;                // WR Schwelle oben

// Alte Werte mit epfangenen Werten verrechnen um zu grosse Evolutionäre
Spruenge zu vermeiden
swMHo = (NswMHo + swMHoAlt) / 2;         // hell
swMHu = (NswMHu + swMHuAlt) / 2;         // dunkel
swMTo = (NswMTo + swMToAlt) / 2;         // warm
swMTu = (NswMTu + swMTuAlt) / 2;         // kalt
swWRu = (NswWRu + swWRuAlt) / 2;         // WR Schwelle unten
swWRo = (NswWRo + swWRoAlt) / 2;         // WR Schwelle oben




// Lebenzeit wieder hoch setzen - Evozeit auf 0 setzen
lebenszeit = 17280000 + millis ();
// Evozeit immer auf 0 falls nicht von 55/wecken neu gesetzt
evozeit = 0;
}                                        // Ende Gesundheit

delay (random (100,500));                // Desynchronisierung
modWach (0,0);                           // gehe in WR Modus
}                                        // Ende Evozeit
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

noch Haupt 2

```
// wenn gesund bzw außerhalb der Evozeit - mach einfach weiter
else {
delay (random (100, 500));              // Desynchronisation
// gehe  in WR Modus mit neuer Energie 10.000
modWach (5000,5000);
}
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
} // Ende if 66/Evo


// Temp, Hell aussen ausreichend
if (sig00 == 77) {
//  hier egal ob Temp oder Hell hoch sind da im Winter sowiso zu kalt
funk77 = 77;
modWach(2000,2000);                     // 4000 drauf wenn aussen ok
}

// Personen anwesend
if (sig00 == 33) {                      // Nutzer anwesend
sigG = 30000;                           // 30.000 - wacht sonst bei Minusgraden
nicht auf
funk33 = 33;
}
else {sigG = 0;}

// Mobileinheit
// Stattsmitglied anwesend, hier keine Aktion gesetzt
if (sig00 == 44) {
funk44 = 44;
sigG2 = 1000;                           // für die Gesundheit als positives Signals
}
else {sigG2 =0;}
// Mobil: zu dunkel -> Messwert Dunkelheit senken - Fenster öffen sich
if (sig00 == 20) {
funk20 = 20;                            // Variable für serielle Ausgabe
mobHell = 2000;
servoPot = servoPot - 5000;             // überdrücke Dämpfung
servowait = 5000;                       // Klappen bleiben länger zu
}

// Mobil: zu hell -> Messwert Helligkeit steigern - Fenster schließen sich
if (sig00 == 21) {
funk21 = 21;
mobHell = -2000;
servoPot = servoPot - 5000;
servowait = 5000;
}
```

noch Haupt 2

```
// Mobil: zu hell -> Messwert Helligkeit steigern - Fenster schließen sich
if (sig00 == 31) {
funk31 = 31;
mobTemp = 100;
servoPot = servoPot - 5000;
servowait = 5000;
}

// Anfrage -Ping Empfangen - Anworte mit Antwort-Ping
if (sig00 == 11) {
funk11 = 11;
// Funk Anworten mit 12
funk (12,zufall,12,12,12,12,12, 1);
funk112 = 112;
}

/ / Antwort-Ping Empfangen
if (sig00 == 12) {
funk12 = 12;
pingalt = pingneu;          // speichere vorhergende Ping-Kennung ab
pingneu = sig01;            // lese neue Ping-Kennung ein
// Wenn zwei verschiedene Pings innerhalb des Zeitfensters dann steige aus RW
Anpassung aus

// pingalt wird auf 0 gesetzt, wenn Zeitfenster abgelaufen ist, muss zurückge-
setzt werden - sonst scheint es immer ein vorhergehendes Ping zu geben obwohl
aus vorheriger RW Anpassung
if ((pingneu != pingalt) && (pingalt != 0)) {
rwflag = 0;                 // Reichweitenan passung zuende
modWach (0,0);              // einfach weitermachen
}                           // Ende pingneu
}                           // Ende  1 2
}                           // Ende Haupt 2
```

Apellmodus

Wecksignal (55) an andere Einheiten senden

```
// Funktionen Modi

// Appell - leitet den Appell nur ein, weckt die anderen, dann echter Appell mit
funk 11, dann Evolutionszeit
void apell (){
//andere wecken
funk (55,55,55,55,55,55,55, 10);
print155 = 155;
// geh erstmal in den Wachmodus bis ein Wecksignal zurückkommt
modWach (1000,1000);        // kleiner Kick zum Aufwachen +2000
gesendet = 55;
} // Ende Appell
```

Wach- Ruhemodus

```
// Wach / Ruhe
void modWach (int sigHell, int sigTemp) {
```

Lokale Variablen setzen
> default keine Aktion

```
// Rueckgabewerte Aktion
int aHWert = 0;                    // Aktionswert Hell
int aTWert = 0;                    // Aktionswert Temp
```

Lokale Variablen setzen
> aus Sensorfunktion auslesen

```
// Sensoren auslesen und harmonisieren nach Modus Schwellenwerten
int modHell = sensorHell (1,swMHo, swMHu);
int modTemp = sensorTemp (1,swMTo, swMTu);
```

Auswertung der Sensor und Funkdaten
> in Fuzzylogic-Neuron
Potentiale addieren und mit Schwellenwerten vergleichen, wenn sie zwischen oberem und unterem Schwellenwert liegen, aktuellen Modus nicht wechseln

```
// ins Neuron damit
WRpot = WRpot + modHell + modTemp + sigHell + sigTemp + sigG + sigG2 +
aktion;

int modWertAlt = modWRwert;
if (WRpot > swWRo) {modWRwert = 1;}
if (WRpot < swWRu) {modWRwert = -1;}
/ / wenn zwischen den Schwellenwerten lass den Wert wie er ist
if ((WRpot >= swWRu) && (WRpot <= swWRo)) {modWRwert = modWertAlt;}
if (WRpot > 100000) {WRpot = 100000;}      // Werte begrenzen
if (WRpot < -100000) {WRpot = -100000;}
```

Gesundheitsermittlung

```
//Gesundheit
// bleibt oben solange: warm genug, hell genug (tagsüber), genug Personen,
wenig Aktion (Energieverbrauch), Zeit tagsüber abnehmend, nachts zunehmend
(Erholung)
```

Lokale Variablen setzen

```
int HG = 0;                        // Helligkeitswert
int TG = 0;                        // Temperaturwert
int ZG = 0;                        // Zeit Ab/Zunahme
int printGesund =0;
// Messen Sensoren
int sensorHG = analogRead (0);
int sensorTG = analogRead (2);
// Auswerten
```

Neuron 1 Helligkeit

```
if ((sensorHG >= swGHo) && (sensorHG <= swGHu)) {HG = 4;}
else {HG = -4;}
```

Neuron 2 Temperatur

```
if ((sensorTG >= swGTu) && (sensorTG <= swGTo)) {TG = 4;}
else {TG = -4;}
// Zeitanteil
if (flagWR == 1) {ZG = -1;}        // Tagsüber Zeit negativ
else {ZG = 1;}                     // Nachts Zeit positiv
```

Berechnung der Gesundheitspotentials

```
// Potential berechnen
gesundPot = gesundPot + HG + TG + ZG;      // Startwert Gesundheit
if (gesundPot >= 1000000) {gesundPot = 1000000;}
if (gesundPot <= -1000000) {gesundPot = -1000000;}
if (gesundPot <=0) {gesundheit = 0; ledpause = 200;}
```

```
// 0=krank/Empfänger 1=gesund/Spender , LED blinken wenn „krank“
else {gesundheit = 1;}
```

Eigenevolutionsmodus

Falls ein Evolutionsschritt erfolgen sollte, aber keine gesunden Einheiten/Spender neue Werte gesendet haben, werden die eigenen Werte mutiert, um das System auch bei komplett kranken Einheiten zu reaktivieren

```
// Evolution falls keine Spender zur Verfügung stehen = alle sind „krank“
// falls eine 55 zum Wecken empfangen und der Evozeitraum gesetzt wurde
if (evozeit != 0) {
// und falls die Evozeit abgelaufen ist ohne, dass Werte empfangen wurden
if (millis () >= evozeit) {
// eigen Werte mutieren
swMHo = swMHo + random (-5, 5) ;          // hell
swMHu = swMHu + random (-5, 5);           // dunkel
swMTo = swMTo + random (-3, 3);           // warm
swMTu = swMTu + random (-3, 3);           // kalt
swWRu = swWRu + random (-10, 10);         // Schwelle Wach Ruhe unten
swWRo = swWRo + random (-10, 10);
evozeit = 0;                              // Zurücksetzen
}                                         // millie
}                                         // Ende evozeit != 0
```

Serielle Ausgabe zum Loggen

Diese Ausgabe gibt es nur im Versuchssystem, um die Funktionsfähigkeit des Systems nachzuweisen. Im eigentlichen ADRRM würden keine Schnittstellen existieren, die ein Mitloggen von Daten erlauben würden.

```
// Serielle Ausgabe - alle
if (serialFlag == 9999) {

// Zufallszahl auf Sig01 damit bei Reichweitenermittlung zwischen der Antwort
mehrere Module unterschieden werden kann, Wert 0 ist raus - siehe Sig00=12,
hier in der Zeitschleife, damit sich die Kennung nicht zu oft ändert, sonst kann es
zwei unterschiedliche Kenungen vom gleichen Modul für die selbe Anfrage ge-
ben und der Empfänger geht irrtümlich von Anworten unterschiedlicher Module
aus
zufall = random (1, 999);

// WR Pot
int printWRpot = WRpot/100;
Serial .print (36, BYTE );
Serial .print (40, BYTE);
Serial .print (printWRpot, DEC );

Serial .print (36, BYTE );          // kein Funk auch 0 ausgeben wegen Logg
Serial .print (37, BYTE);
Serial .print (funk77, DEC );
funk77=0;                           // zurücksetzen

if (funk11 != 0) {
Serial .print (36, BYTE );
Serial .print (37, BYTE);
Serial .print (funk11, DEC );
funk11=0;
}
```

noch Serielle Ausgabe

```
if (funk12 != 0) {
Serial .print (36, BYTE );
Serial .print (37, BYTE);
Serial .print (funk12, DEC );
funk12=0;
}

if (funk20 != 0) {
Serial .print (36, BYTE );
Serial .print (37, BYTE);
Serial .print (funk20, DEC );
funk20=0;
}

if (funk21 != 0) {
Serial .print (36, BYTE );
Serial .print (37, BYTE);
Serial .print (funk21, DEC );
funk21=0;
}

if (funk31 != 0) {
Serial .print (36, BYTE );
Serial .print (37, BYTE);
Serial .print (funk31, DEC );
funk31=0;
}

if (funk33 != 0) {
Serial .print (36, BYTE );
Serial .print (37, BYTE);
Serial .print (funk33, DEC );
funk33=0;
}

if (funk44 != 0) {
Serial .print (36, BYTE );
Serial .print (37, BYTE);
Serial .print (funk44, DEC );
funk44=0;
}

if (funk55 != 0) {
Serial .print (36, BYTE );
Serial .print (37, BYTE);
Serial .print (funk55, DEC );
funk55=0;
}
```

noch Serielle Ausgabe

```
if (funk66 != 0) {
Serial .print (36, BYTE );
Serial .print (37, BYTE);
Serial .print (funk66, DEC );
funk66=0;
}

if (funk99 != 0) {
Serial .print (36, BYTE );
Serial .print (37, BYTE);
Serial .print (funk99, DEC );
funk99=0;
}
if (funk88 != 0) {
Serial .print (36, BYTE );
Serial .print (37, BYTE);
Serial .print (funk88, DEC );
funk88=0;
}

if ( print155 != 0) {
Serial .print (36, BYTE );
Serial .print (37, BYTE);
Serial .print ( print155, DEC );
print155=0;
}

if ( print166 != 0) {                    // selber 66 gesendet (nicht empfangen)
Serial .print (36, BYTE );
Serial .print (37, BYTE);
Serial .print ( print166, DEC );
print166=0;
}

if ( funk111 != 0) {                     // selber 11 gesendet (nicht empfangen)
Serial .print (36, BYTE );
Serial .print (37, BYTE);
Serial .print ( funk111, DEC );
funk111=0;
}

if ( funk112 != 0) {                     // selber 12 gesendet (nicht empfangen)
Serial .print (36, BYTE );
Serial .print (37, BYTE);
Serial .print ( funk112, DEC );
funk112=0;
}
```

```
// WR Zustand
int printWR = modWRwert *100;              // +100 wach, -100 Ruhe
Serial .print (36, BYTE );
Serial .print (44, BYTE);
Serial .print (printWR, DEC );

// Ausgabe neuer Schwellenwert Hell
- noch eine Serielle Ausgabe frei, statt Logg Lebenszeit
Serial .print (36, BYTE );
Serial .print (45, BYTE);
Serial .print (swMHo, DEC );

// Sendeleitung
int slprint = sl *100;
Serial .print (36, BYTE );
Serial .print (43, BYTE);
Serial .print (slprint, DEC );

// Messwert Sensor Hell
Serial .print (36, BYTE );
Serial .print (38, BYTE);
Serial .print (sensorH, DEC );

// Messwert Sensor Temp
Serial .print (36, BYTE );
Serial .print (39, BYTE);
Serial .print (sensorT, DEC );

// Gesundheitszustand
int printGesund = gesundPot/1000;          // auf sinnvolles Maß kürzen
Serial .print (36, BYTE );
Serial .print (41, BYTE);
Serial .print (printGesund, DEC );

// Lebenszeit
int printLeben = restzeit/100000;          // auf sinnvolles Maß kürzen
Serial .print (36, BYTE );
Serial .print (42, BYTE);
Serial .print (printLeben, DEC );

} // Ende Serial
```

Wachmodus

Blinke mit Led

Flags setzen, um festzustellen, ob die Einheit erwacht ist:
>speichere ab, in welchem Zustand du warst
>setze deinen Zustand auf Wach

```
// Wach //
if (modWRwert == 1) {
digitalWrite (13, HIGH );          // Led wenn wach
delay (ledpause);
digitalWrite (13, LOW );
delay (ledpause);
digitalWrite (13, HIGH );
flagWRalt = flagWR;
flagWR = 1;
```

Anpassung der Sendereichweite

diese Funktion überprüft zu einem zufällig gewählten Zeitpunkt, ob sich mindesten zwei weitere Einheiten in Reichweite befinden.
Dies geschieht zunächst mit der geringsten Sendeleitung und wird so lange gesteigert, bis sich zwei Einheiten zurückgemeldet haben oder die maximale Sendeleistung erreicht ist.

```
// Reichweitenanpassung
// Zufallszeitpunkt erzeugen und starten, der Rest passiert oben bei sig00 = 12
int zfl = random (1, 5000);
// wenn Zufallszahl = 3 (oder andere Zahl im Zahlenbereich) und RWanpassung
nicht schon läuft, dann Pinge
if ((zfl == 3) && (rwflag == 0)) {
pingzeit = 0;                      // starte die Ping - Zeit
sl = 1;                            // Sendeleistung auf Minimalwert starten
Radio.power = sl;                  // Sendeleitung setzen
funk (11,11,11,11,11,11,11, 1);    // Ping senden
funk111 = 111;                     // selber 11 gesendet (nicht empfangen)
rwflag = 1;                        // Reichweitenanpassung aktiv
}

// falls keine Anwort innerhalb der Ping-Zeit reinkommt, Sendeleitung in zwei
Schritten steigern
 // falls RWAnpassung aktiv ist u. keine Antwort innerhalb der Ping-Zeit reinkam
if ((rwflag == 1) && (pingzeit >= 99)) {
pingzeit = 0;                      // Ping - Zeit neu starten
pingalt = 0;                       // Empfangene Ping zurücksetzen
sl = sl++;                         // Sendeleitungslsteigern
if (sl >= 3) {
sl=3;                              // bis max. 3
rwflag = 0;                        // Ende RW Anpassung
}

Radio.power = sl;                  // Sendeleitung neu setzen
funk (11,11,11,11,11,11,11, 1);    // Ping senden
funk111 = 111;                     // selber 11 gesendet ( n i c ht empfangen)
rwflag = 1;                        // Reichweitenanpassung läuft
}                                  // Ende rw

// pingzeit steigern bis max 100
if (pingzeit <= 100) {
pingzeit = pingzeit++;
}
```

noch Wachmodus

Schwellenwerte werden bei jeder Einheit leicht variiert um die Schwellenwertband-breite zu erhalten -> Evolution

```
// sw-Schwellenwert, A-Aktion, H-Hell, o-oben / u-unten
int swAHu = 600 + random (-10, 10);          // wenn dunkler als 700 - auf
int swAHo = 300 + random (-5, 5);            // wenn heller (kleiner) als 300 - zu
int swATu = 38 + random (-2, 2);             // kleiner als - Lueftung zu / Winter!
int swATo = 90 + random (-5, 5);             // größer als - Lüftung auf
```

Aktion ausführen

Werden die Schwellenwerte über- bzw. unterschritten werden die Servos in die entsprechenden Positionen gefahren.
Die „aktion“ Variable zieht bei Servobedie-nung Energie ab - > Energiemanagement

```
// Aktion
aHWert = sensorHell (2,swAHo, swAHu);
aTWert = sensorTemp (2,swATo, swATu);
if (aHWert == -1) { servoFenAuf(); }
if (aHWert == 1) { servoFenZu(); }
if (aTWert == 1) { servoLuefAuf(); }
if (aTWert == -1) { servoLuefZu(); }
// wenn keine Aktion nichts bei Gesundheit abziehen
if (aTWert == 0) {aktion = 0; }
if (aHWert == 0) {aktion = 0; }
}                                            // Ende Wach
```

Ruhemodus

```
// Ruhe
if (modWRwert == -1) {
digitalWrite (13, LOW );
flagWRalt = flagWR;
flagWR = 0;
```

Schwellenwerte werden bei jeder Einheit leicht variiert um die Schwellenwertband-breite zu erhalten -> Evolution

```
// sw-Schwellenwert, A-Aktion, H-Hell, o-oben / u-unten
int swAHu = 1040 + random (-5, 5);           // immer zu nachts
int swAHo = 800 + random (-5, 5);            // kleiner als - Fenster zu
int swATu = 50 + random (-5, 5);             //wenn heiß - Lüftung auf
int swATo = 95 + random (-5, 5);             // kleiner als - lauwarm - Lüftung zu
```

Aktion ausführen

Werden die Schwellenwerte über- bzw. unterschritten werden die Servos in die entsprechenden Positionen gefahren.
Fehler-unkritisch, die „aktion“ Variable wird nicht gesetzt!

```
// Aktion
aHWert = sensorHell (3,swAHo, swAHu);
aTWert = sensorTemp (3,swATo, swATu);
if (aHWert == 1) { servoFenZu(); }
if (aHWert == -1) { servoFenAuf(); }
if (aTWert == 1) { servoLuefAuf(); }
if (aTWert == -1) { servoLuefZu(); }

} // Ende Ruhe
} // Ende Wach / Ruhe
```

```
// Funktionen Dienende ++++++++++++++++++++++++++++++++++++++++++++++++
```

Lebenszeitbegrenzung

Wenn die Lebenszeit abgelaufen ist, wird die Einheit auf krank gesetzt und bekommt beim nächsten Evolutionären Zyklus neue Werte -> Systemdynamik erhalten

```
// Lebensezeitbegrenzung
void leben () {
restzeit = lebenszeit - millis ();
if (restzeit <=0) { lebenszeit = 0; gesundheit = 0; }
}
```

Sensor Helligkeit auslesen Fuzzy-Neuron

Der Helligkeitssensor am analog-in-Pin Nr.0 wird ausgelesen und ein Gefahrenwert draufgerechnet - falls vorhanden.

Da diese Funktion von mehreren anderen Funktionen aufgerufen werden kann, werden die Werte in verschiedenen Variablen abgespeichert. Die alten Werte werden vorher in ...alt(x) gesichert

```
// Sensor Hell
int sensorHell(int aufruf, int swHo, int swHu) {
sensorH = analogRead (0) + mobHell;          // Mobil- Gefahrenwert addieren
int harmH = 0;
int harmHalt1 = 1;
int harmHalt2 = 1;
int harmHalt3 = 1;
if (aufruf == 1) {harmHalt1 = harmH1;}       // speichert aktuellen Wert ab
if (aufruf == 2) {harmHalt2 = harmH2;}       // speichert aktuellen Wert ab
if (aufruf == 3) {harmHalt3 = harmH3;}       // speichert aktuellen Wert ab
if (sensorH <= swHo){harmH = 1;}   // es ist hell - SW oben 300/900 -zu
if (sensorH >= swHu){harmH = -1;}  // es ist dunkel - SW unten 700/1020 -auf
if ((sensorH > swHo) && (sensorH < swHu)) {
if (aufruf == 1) {harmH = harmHalt1;}        // lasse Klappen wo sie sind
if (aufruf == 2) {harmH = harmHalt2;}        // lasse Klappen wo sie sind
if (aufruf == 3) {harmH = harmHalt3;}        // lasse Klappen wo sie sind
}
if (mobHell > 0) {mobHell–;}                 // Mobilwert zurücksetzen
if (mobHell < 0) {mobHell++;}
if (mobHell == 0) {mobHell = 0;}
if (servoPot > 0) {servoPot–;}               // senkt ServoPot
if (servoPot <= 0) {servoPot = 0;}
if (aufruf == 1) {int harmH1 = harmH;}       // speichert aktuellen Wert ab !
if (aufruf == 2) {int harmH2 = harmH;}       // speichert aktuellen Wert ab !
if (aufruf == 3) {int harmH3 = harmH;}       // speichert aktuellen Wert ab !
// brauche ich das wenn globale varialble geändert???
return harmH;
}
```

Sensor-Temperatur auslesen Fuzzy-Neuron

Der Helligkeitssensor am analog-in-Pin Nr.0 wird ausgelesen und ein Gefahrenwert draufgerechnet - falls vorhanden.

```
// Sensor Temp
// modTemp = sensorTemp (swMTo, swMTu);
int sensorTemp(int aufrufT, int swTo, int swTu) {
sensorT = analogRead (2) + mobTemp; // Mobilwert oder Gefahr
int harmT = 0;
int harmTalt1 = 1;
int harmTalt2 = 1;
int harmTalt3 = 1;
if (aufrufT == 1) {int harmTalt1 = harmT1;} // speichert aktuellen Wert ab
if (aufrufT == 2) {int harmTalt2 = harmT2;} // speichert aktuellen Wert ab
if (aufrufT == 3) {int harmTalt3 = harmT3;} // speichert aktuellen Wert ab
```

noch Sensor Temperatur auslesen

```
int harmTalt = harmT;                          // speichert aktuellen Wert ab
if (sensorT <= swTu){ harmT = -1;}             // kalt - Schwelle unten
if (sensorT >= swTo){harmT = 1;}               // warm - Schwelle oben
if ((sensorT > swTu) && (sensorT < swTo)) {
if (aufrufT == 1) {harmT = harmTalt1;}         // lasse Klappen wo sie sind
if (aufrufT == 2) {harmT = harmTalt2;}         // lasse Klappen wo sie sind
if (aufrufT == 3) {harmT = harmTalt3;}         // lasse Klappen wo sie sind
}

if (mobTemp > 0) {mobTemp–;}                   // Mobilwert zurücksetzen
if (mobTemp < 0) {mobTemp++;}
if (mobTemp == 0) {mobTemp = 0;}
if (servoPotTemp > 0) {servoPotTemp–;}         // senkt ServoPot
if (servoPotTemp <= 0) {servoPotTemp = 0;}
return harmT;
}
```

Funk - senden

Es können sechs verschiedene Werte übergeben werden, der erste Wert gibt den Signaltyp an -> Gefahr, Sensor. Die weiteren Werte werden nur zur Übermittlung von Schwellenwerten während der Evolutions-zeit genutzt.

Fall der neu übergebene Signaltyp „sigy00“ dem zuletzt gesendeten „gesendet“ entspricht, wird er nicht weitergeleitet um Singalrundläufe zu vermeiden.

```
// FUNK ++++++++++++++++++++++++++++++++++
void funk (int sigy00, int sigy01,int sigy02,int sigy03,int sigy04,int sigy05,int
sigy06,int wiederholung) {
// wenn der erste empfangene Wert/Nachrichtentyp sigy00 gleich dem letzten
gesendeten, dann kein Funk da wahrscheinlich Rundlauf
if (sigy00 == gesendet) {
}
else {
for (int w=0; w <= wiederholung; w++) {
Radio.txMode(7);
Radio.data[0] = sigy00;
Radio.data[1] = sigy01;
Radio.data[2] = sigy02;
Radio.data[3] = sigy03;
Radio.data[4] = sigy04;
Radio.data[5] = sigy05;
Radio.data[6] = sigy06;
Radio.write ();
}                                              // for
}                                              // else
Radio.rxMode(7);                               // auf Empfang, 7 byte
}                                              // nichts zurückzugeben
```

Servos-Anschläge definieren

Hier werden die maximalen Positionen der Servos festgelegt. Die Dämpfung verhindert ein hyperaktives Verhalten der Servos.

```
// SERVO BEWEGUNGSVOREINSTELLUNGEN//
// wenn Dämpfung zuende dann fahre
void servoFenZu() {if (servoPot < 1) {servo1 ( 0, 140, 45, 40);}}
void servoLuefZu() {if (servoPotTemp < 1) {servo2 ( 0, 120, 52, 40);}}
void servoFenAuf() {if (servoPot < 1) {servo1 ( 1, 145, 45, 40);}}
void servoLuefAuf() {if (servoPotTemp < 1) {servo2 ( 1, 125, 52, 40);}}
```

noch Servos-Anschläge definieren

```
/*
// SERVO BEWEGUNGSVOREINSTELLUNGEN Modul 7
// wenn Dämpfung zuende dann fahre
void servoFenZu() {if (servoPot < 1) {servo1 ( 0, 140, 45, 40);}}
void servoLuefZu() {if (servoPotTemp < 1) {servo2 ( 1, 130, 65, 40);}}
void servoFenAuf() {if (servoPot < 1) {servo1 ( 1, 145, 45, 40);}}
void servoLuefAuf() {if (servoPotTemp < 1) {servo2 ( 0, 130, 65, 40);}}
*/
```

Servos fahren

Dieser Programmteil basiert auf der Servosteuerung von Ralph Zimmermann.

Die Servos könnten dirket in die jeweiligen Positionen gefahren werden allerdings bewegen sie sich dann so schnell, dass die Klappen beschädigt wurden. Es musste daraufhin eine Lösung gefunden werden, die Servos in einer kontrollierten Geschwindigkeit zu fahren, was sich insofern als schweirig erwies, da die verwendeten analogen Servos keine Positionsrückmeldungen geben.

```
// SERVOS - basiert auf der Servosteuerung von Ralph Zimmermann ;o)
// Fenster
void servo1 (int auf, int aufschlag, int zuschlag, int geschwind) {
int printHServo =0;
if (auf == 1){endpos=aufschlag; printHServo = 50;}
else {endpos=zuschlag; printHServo = -50;}
//setze drehrichtung
if (endpos<thispos){vz= -1;}                    // schrittweise auf und warten
else{vz=1;}                                     // nicht warten Notfall sofort zu
//fahre
while (thispos!=endpos){
servoPot = 8000+ random (-100, 200) + servowait; // Starte Dämpfung mit Zufall
servowait = 0;                                   // zurücksetzen
thispos=thispos+1*vz;
myservo1.write (thispos);
delay(geschwind);                                // geschwindigkeit steuern
}
aktion = -1;
}                                                // Ende Servo1

// Lueftung
void servo2 (int auf, int aufschlag, int zuschlag, int geschwind) {
int printTServo = 0;
if (auf == 1){endpos2=aufschlag; printTServo = 60;} // anders als Hell damit die
Linien im Grapher nicht übereinander
else {endpos2=zuschlag; printTServo = -60;}
//setze Drehrichtung
if (endpos2<thispos2){vz=-1;}                   // schrittweise auf und warten
else{vz=1;}                                     // nicht warten Notfall sofort zu
//fahre
while (thispos2!=endpos2){
servoPotTemp = 8000+ random (-500, 500) + servowait;  // Dämpfung mit Zufall
servowait = 0;                                  // zurücksetzen
thispos2=thispos2+1*vz;
myservo2.write (thispos2);
delay(geschwind); //hiermit die geschwindigkeit steuern
}
aktion = -1;
} // Ende Servo // ADRRM Hub
```

ADRRM-Progammierung der Überbrückungseinheiten

mit bereinigten Kommentaren

Bibliotheken aufrufen
Variablen definieren

```
// die Hubs überbrücken Lücken ziwschen den Steuerungseinheiten, sie selbst steuern nichts, sie funken immer mit höchster Sendeleistung
// da sie im Versuchsaufbau größere Distanzen überbrücken müssen, im Regelbetrieb würde man sie jedoch so dicht setzen, dass sie
// Sendeleistungsreserven haben und daher auch eine Anpassung der Sendeleistung vornehmen könnten

#include <Nrf2401.h>        // Transceiver/Funk Bibliothek
        Nrf2401 Radio;


// Radio
        int sig00 = 0;              // Art der Nachricht - Gefahr, Schwellenwerte etc.

// Uebertragene Werte - werden nur bei der Übertragung von neuen Schwellenwerten im Evolutionsmodus benötigt
        int sig01 = 0;
        int sig02 = 0;
        int sig03 = 0;
        int sig04 = 0;
        int sig05 = 0;
        int sig06 = 0;

        int anzahl = 1;         // Anzahl der Weiterleitungen - normalerweise auf 1

        int gesendet = 0;
        int zufall = random(1, 999);    // Zufallsvariable für Reichweitenregulierung
        int timer = 0;
```

Voreinstellungen

Alle Einheiten im gesamten System haben die gleiche Adresse „1“ - somit ist, wie auch gewünscht, keine Adressierung einzelner Einheiten möglich !!

```
void setup (){                      // Transceiver / Funk initialisieren
      Radio.power = 3;              // Sendeleitung am Anfang auf max. = 3
      Radio.localAddress = 1;       // eigene Adresse - alle identische Adressen
      Radio.remoteAddress = 1;      // von Sender - alle identische Adressen
      Radio.rxMode(7);              // Empfangsbereit für 7 Byte
}
```

Schleife

solange nichts empfangen wird, tue nichts

```
void loop (){
   while(!Radio.available()) {      // solange keine Signale empfangen werden:
                                    // Led auslassen, ansonsten nichts tun
     analogWrite(10, 0);
     analogWrite(9, 0);
     analogWrite(11, 0);
       }
```

Falls Daten empfangen werden

Sende sie weiter, bis auf einige Ausnahmen

```
// bei Empfang

// Zufallskennung erzeugen und einige Zeit beibehalten um nicht
zweimal hintereinander unterschiedliche Kennungen zu senden, die
als zwei Nachbarn gedeutet würden
if (timer >= 1000) {
zufall = random(1, 999);
timer = 0;
}

// empfangene Daten abspeichern
  Radio.read();                 // Daten aus Empfangspuffer auslesen
  sig00 = Radio.data[0];        // Signaltyp - Gefahr, Schwellenwerte etc.
  sig01 = Radio.data[1];        // weitere Uebertragene Werte
  sig02 = Radio.data[2];
  sig03 = Radio.data[3];
  sig04 = Radio.data[4];
  sig05 = Radio.data[5];
  sig06 = Radio.data[6];



// Notfallsignale (99 Eindringling, 88 Wind) mehrere Male weiterleiten,
andere nur einmal
  if ((sig00 == 99) || (sig00 == 88)) {
  anzahl = 10;

  analogWrite(9,  0);           // Led einschalten
  analogWrite(10, 0);
  analogWrite(11, 200);
  }

// andere Signale nur einmal weiterleiten
  else {
  anzahl = 1;

  analogWrite(9,  0);           // Led mit anderer Farbe einschalten
  analogWrite(10, 200);
  analogWrite(11, 0);
  }

  // falls Reichweiten-Ping empfangen antworte mit 12
  if (sig00 == 11) {
   sig00 = 12;
   sig01 = zufall;
  }
```

```
 // Signale von Mobilmodul nicht weiterleiten diese sollen nur lokal wirken
      if ((sig00 != 31) && (sig00 != 21) && (sig00 != 20)){

// andere Signale an Sendefunktion übergeben, anzahl = Wiederholungen
      funk (sig00,sig01,sig02,sig03,sig04,sig05,sig06, anzahl);
      }

} // Ende loop


// FUNK ++++++++++++++++++++++++++++++++++
// Sendefunktion

 void funk (int sigy00, int sigy01,int sigy02,int sigy03,int sigy04,int sigy05,int
sigy06,int wiederholung) {

// wenn der erste empfangene Wert/Nachrichtentyp sigy00 gleich dem letzten
gesendeten , dann nicht weiterleiten, da wahrscheinlich Rundlauf

          if (sig00 != gesendet) {

          for (int w=0; w <= wiederholung; w++) {

           Radio.txMode(7);
             Radio.data[0] = sigy00;
             Radio.data[1] = sigy01;
             Radio.data[2] = sigy02;
             Radio.data[3] = sigy03;
             Radio.data[4] = sigy04;
             Radio.data[5] = sigy05;
             Radio.data[6] = sigy06;
             Radio.write();

             gesendet = sigy00;

 // kleine Pause, sonst nützen die Wiederholungen nichts, er sendet dann einfach
zu schnell hintereinander weg (Shockburst enabeld)
             delay (100);

           } // for
          } // if

        Radio.rxMode(7); // gehe wieder auf Empfang, 7 byte

       } // nichts zurückzugeben
```

ADRRM-Progammierung der äußeren Sensoreinheiten - Variante ohne Weiterleitung

mit bereinigten Kommentaren

Bibliotheken aufrufen
Variablen definieren

```
#include <Nrf2401.h>        // Transceiver/Funk Bibliothek
Nrf2401 Radio;

// Radio
      int sig00 = 77;            // Art der Nachricht - Gefahr, Schwellenwerte etc.

// Uebertragene Werte - werden nur bei der Übetragung von neuen Schwellen-
werten im Evolutionsmodus benötigt
      int sig01 = 77;
      int sig02 = 77;
      int sig03 = 77;
      int sig04 = 77;
      int sig05 = 77;
      int sig06 = 77;

      int anzahl = 1;
      unsigned long millis();
      unsigned long sensorZeit = millis() + 500;
```

Voreinstellungen

Alle Einheiten im gesamten System haben die gleiche Adresse „1“ - somit ist, wie auch gewünscht, keine Adressierung einzelner Einheiten möglich !!

```
void setup (){
 // Transceiver / Funk initialisieren
      Radio.power = 2;              // Sendeleitung am Anfang auf mittel = 2
      Radio.localAddress = 1;       // eigene Adresse - alle identische Adressen
      Radio.remoteAddress = 1;      // von Sender - alle identische Adressen
      Radio.txMode(7);              // Sendebereit für 7byte
}
```

Schleife

Messe die Außenhelligkeit und sag den inneren EinheitenBescheid falls es hell genug ist.

```
void loop (){
    if (sensorZeit < millis()) {                  // und Sensorlesepause zuende

    sensorZeit = millis() + 500;                  // setze neue Pausenzeit
    int sensorHell = analogRead (0);              // lese Sensoren aus
    int sensorTemp = analogRead (3);

    analogWrite(10, 0);                           // schalte Led auf „standby“ Farbe
    analogWrite(9, 50);
    analogWrite(11, 0);

    if (sensorHell <=980) {                           // und wenn es Hell genug ist
 // Funke dies den Modulen innen zu
    funk (sig00,sig01,sig02,sig03,sig04,sig05,sig06, anzahl);
```

```
    analogWrite(10, 200);                    // blinke in einer Farbe
    delay (100);
    analogWrite(10, 0);

    } // Ende sensorHell
    } // Ende sensorZeit

} // Ende loop


// FUNK ++++++++++++++++++++++++++++++++++++

void funk (int sigy00, int sigy01,int sigy02,int sigy03,int sigy04,int sigy05,int
sigy06,int wiederholung) {

        for (int w=0; w <= wiederholung; w++) {
         Radio.txMode(7);
           Radio.data[0] = sigy00;
           Radio.data[1] = sigy01;
           Radio.data[2] = sigy02;
           Radio.data[3] = sigy03;
           Radio.data[4] = sigy04;
           Radio.data[5] = sigy05;
           Radio.data[6] = sigy06;
           Radio.write();
          }                                  // for


      }                                      // nichts zurückzugeben
```

ADRRM-Progammierung der äußeren Personendetektionseinheiten

mit bereinigten Kommentaren

Bibliotheken aufrufen
Variablen definieren

```
/*  Bei PIR Aktivitaet Wert=99 senden
 Funktioniert - aber nur mit der seltsamen Pause für den PIR
 */

#include „Nrf2401.h“
Nrf2401 Radio;
int forMax = 0;                          // blink Anzahl
int PIR;
int sig1 = 0;
int sig2 = 0;
int sig3 = 0;
int sig4 = 0;
int sig5 = 0;
int sig6 = 0;
int sig7 = 0;

int radioAlt = 0; // letzter gesendeter Wert

unsigned long millis();
int sensorZeit = millis() + 1000;
```

Voreinstellungen

Alle Einheiten im gesamten System haben die gleiche Adresse „1“ - somit ist, wie auch gewünscht, keine Adressierung einzelner Einheiten möglich !!

```
void setup(void)
{
 Radio.localAddress = 1;                 // eigene Adresse
 Radio.remoteAddress = 1;                // Empfänger Adresse
 Radio.txMode(7);                        // 7 byte senden
}
```

Schleife

Lese den Passiv-Infrarot-Detektor aus um festzustellen ob ein Lebewesen anwesend ist.

```
void loop(void)
{
  delay (10);                            // seltsame PIR-Pause
  PIR = analogRead(2);                   //lese PIR aus
  int sensorHell = analogRead (0);
  int sensorTemp = analogRead (3);

// PIR Detektion -  Einheit 6 muss hier <2 stehen, Kontaktprobleme
  if (PIR <= 1) {
   Radio.txMode(7);                      //  99 Gefahren senden

   Radio.data [0] = 99;
   Radio.data [1] = 99;
   Radio.data [2] = 99;
   Radio.data [3] = 99;
   Radio.data [4] = 99;
```

```
    Radio.data [5] = 99;
    Radio.data [6] = 99;
    Radio.write();                  // raus in den Aether mit dem Kram

    analogWrite(11, 200);           // blinke in einer Farbe
    delay (50);                     // kurz anlassen, dann abschalten

    analogWrite(10, 0);
    analogWrite(9, 0);
    analogWrite(11, 0);
   }                                // fertig mit senden //


   if (sensorZeit < millis()) {     // wenn Sendepause abgelaufen

// Helligkeit außen messen
     if ((sensorHell <=1000) || (sensorTemp <= 600)) {

    Radio.txMode(7);                // auf senden gehen und 77 senden

    Radio.data [0] = 77;             // 77 = Helligkeit im Toleranzbereich
    Radio.data [1] = 77;
    Radio.data [2] = 77;
    Radio.data [3] = 77;
    Radio.data [4] = 77;
    Radio.data [5] = 77;
    Radio.data [6] = 77;
    Radio.write();                  // raus in den Aether mit dem Kram //
    // fertig mit senden //

    analogWrite(10, 200);           // blinke in einer Farbe //
    delay (100);                    // kurz anlassen, dann abschalten //
    analogWrite(10, 0);
    analogWrite(9, 0);
    analogWrite(11, 0);

// Sendepause, Zeitraum neu starten
    sensorZeit = millis() + 500;
   }                                // ende if //
   }                                // sensorZeit

// Led auschalten
    analogWrite(10, 0);
    analogWrite(9, 0);
    analogWrite(11, 0);

 }                                  // LOOP
```

ADRRM-Progammierung der inneren Personendetektionseinheiten

mit bereinigten Kommentaren

Bibliotheken aufrufen
Variablen definieren

```
/*  Bei PIR Aktivitaet Wert=44 senden
 Funktioniert - aber nur mit der seltsamen Pause für den PIR
 */

#include „Nrf2401.h“
Nrf2401 Radio;
int forMax = 0;                    // blink Anzahl
int PIR;
int sig1 = 0;
int sig2 = 0;
int sig3 = 0;
int sig4 = 0;
int sig5 = 0;
int sig6 = 0;
int sig7 = 0;

int radioAlt = 0;                  // letzter gesendeter Wert
unsigned long millis();
int sensorZeit = millis() + 1000;
```

Voreinstellungen

Alle Einheiten im gesamten System haben die gleiche Adresse „1“ - somit ist, wie auch gewünscht, keine Adressierung einzelner Einheiten möglich !!

```
void setup(void)
{
 Radio.localAddress = 1;           // eigene Adresse //
 Radio.remoteAddress = 1;          // von Sender //
 Radio.rxMode(7);                  // 7 byte senden //
}
```

Schleife

lese den Passiv-Infrarot-Detektor aus um festzustellen ob ein Lebewesen anwesend ist

```
void loop(void)
{
 Radio.rxMode(7);

 while(!Radio.available()) {

   PIR = analogRead(2);            //lese PIR aus //

// wenn der PIR Detektion
   if (PIR == 1) {
    Radio.txMode(7);               // auf senden gehen und 33 senden //
    Radio.data [0] = 33;           // 33 heißt Person innen detektiert
    Radio.data [1] = 33;
    Radio.data [2] = 33;
    Radio.data [3] = 33;
    Radio.data [4] = 33;
    Radio.data [5] = 33;
```

```
    Radio.data [6] = 33;
    Radio.write();                  // raus in den Aether mit dem Kram

    analogWrite(10, 200);           // blinke in einer Farbe
    delay (500);                    // kurz anlassen, dann abschalten
    analogWrite(10, 0);
    analogWrite(9, 0);
    analogWrite(11, 0);

    Radio.rxMode(7)
   }                                // fertig mit senden
  }                                 // While


// wenn was reinkommt dann auf Empfang gehen
  Radio.read();
  analogWrite(11, 250);             // zeige mit Led an, dass was reinkommt
  delay (200);
  analogWrite(11,0);

  sig1= Radio.data[0];
  sig2= Radio.data[1];
  sig3= Radio.data[2];
  sig4= Radio.data[3];
  sig5= Radio.data[4];
  sig6= Radio.data[5];
  sig7= Radio.data[6];

// Empfangene Singale weiterleiten
  Radio.txMode(7);
  Radio.data[0] = sig1;
  Radio.data[1] = sig2;
  Radio.data[2] = sig3;
  Radio.data[3] = sig4;
  Radio.data[4] = sig5;
  Radio.data[5] = sig6;
  Radio.data[6] = sig7;
  Radio.write();

  // gehe wieder auf Empfang
  Radio.rxMode(7);

  analogWrite(10, 0);
  analogWrite(9, 0);
  analogWrite(11, 0);


 }                                  // LOOP
```

ADRRM-Progammierung der mobilen Einheiten

mit bereinigten Kommentaren

Bibliotheken aufrufen
Variablen definieren

```
/*  ADRRM Mobil Beta 25.Oktober 2010
Funktionen Fenster zu, Lüftung auf, da im Herbst/Winter normalerweise Fenster
auf und Lüftung zu und nur 2 Schalter zur Verfügung stehen
 */

#include „Nrf2401.h“
Nrf2401 Radio;

int sig1 = 0;
int sig2 = 0;
int sig3 = 0;
int sig4 = 0;
int sig5 = 0;
int sig6 = 0;
int sig7 = 0;

int forMax = 0;                 // blink Anzahl
int gruen = 0;
int rot = 0;                    // (hat Wackelkontakt)
int schwarz = 0;
```

Voreinstellungen

Alle Einheiten im gesamten System haben die gleiche Adresse „1“ - somit ist, wie auch gewünscht, keine Adressierung einzelner Einheiten möglich !!

```
void setup(void)

{
 Radio.localAddress = 1;        // eigene Adresse
 Radio.remoteAddress = 1;       // von Sender
 Radio.rxMode(1);               // 1 Byte senden
 Radio.power = 2;               // Sendeleistung begrenzen
 pinMode (12, INPUT);
 pinMode (8, INPUT);

 digitalWrite (12, HIGH);
 digitalWrite (8, HIGH);
}
```

Schleife

lese die Taster aus und sende die dementsprechenden Signale

```
void loop(void)
{
 Radio.rxMode(1);               // jetzt gehe auf Empfang
 while(!Radio.available()) {
  // solange kein Empfang, tue das, was jetzt folgt
   gruen = digitalRead(12);     //lese Taster aus
   rot = digitalRead(8);
   // schwarz = analogRead(2);  // rausgenommen, geht nicht gut
```

```
  if (gruen == LOW) {       // grüne Taste gedrueckt - zu Hell= Fenster zu

    analogWrite(10, 0);     // blinke in einer Farbe
    analogWrite(9, 200);    // mache die anderen aus, falls sie noch an sind
    analogWrite(11, 0);
    delay (100);            // kurz anlassen, dann abschalten
    analogWrite(10, 0);

// auf Senden gehen
    Radio.txMode(7);
  Radio.data[0] = 21;
  Radio.data[1] = 21;
  Radio.data[2] = 21;
  Radio.data[3] = 21;
  Radio.data[4] = 21;
  Radio.data[5] = 21;
  Radio.data[6] = 21;
  Radio.write();
    // fertig mit senden //
    Radio.rxMode(7);        // wieder auf Empfang gehen, 7 byte
  }                         // ende if

     if (rot == LOW) {      // wenn rote Taste gedrueckt: zu warm=Lüftung zu

    analogWrite(11, 200);   // blinke in einer Farbe
    delay (100);            // kurz anlassen, dann abschalten
    analogWrite(10, 0);

// auf Senden gehen
    Radio.txMode(7);
  Radio.data[0] = 31;
  Radio.data[1] = 31;
  Radio.data[2] = 31;
  Radio.data[3] = 31;
  Radio.data[4] = 31;
  Radio.data[5] = 31;
  Radio.data[6] = 31;
  Radio.write();

    // fertig mit senden
    Radio.rxMode(7);        // wieder auf Empfang gehen
  }                         // ende if

  /* schwarzer Taster rausgenommen wegen Wackelkontakt
     if (schwarz >= 300) { // wenn schwarze Taste gedrueckt

    analogWrite(10,0); // blinke in einer Farbe //
    analogWrite(9, 0); // und mache die anderen aus, falls sie noch an sind //
```

```
    analogWrite(11, 200);
    delay (100); // kurz anlassen, dann abschalten //
    analogWrite(11, 0);

    Radio.txMode(7);
  Radio.data[0] = 99;
  Radio.data[1] = 99;
  Radio.data[2] = 99;
  Radio.data[3] = 99;
  Radio.data[4] = 99;
  Radio.data[5] = 99;
  Radio.data[6] = 99;
  Radio.write();
    // fertig mit senden //

    Radio.rxMode(7); // wieder auf Empfang gehen //

   } // ende if
*/
    analogWrite(10, 0);
    analogWrite(9, 0);
    analogWrite(11, 0);
 }

 Radio.read();                  // wenn was reinkommt, dann auf Empfang gehen
  sig1= Radio.data[0];
  sig2= Radio.data[1];
  sig3= Radio.data[2];
  sig4= Radio.data[3];
  sig5= Radio.data[4];
  sig6= Radio.data[5];
  sig7= Radio.data[6];
  analogWrite(11, 50);        // zeige mit Led an, dass was reinkommt

  // Weiterleiten - nicht nötig, bleibt aber mal drin
  Radio.txMode(7);
  Radio.data[0] = sig1;
  Radio.data[1] = sig2;
  Radio.data[2] = sig3;
  Radio.data[3] = sig4;
  Radio.data[4] = sig5;
  Radio.data[5] = sig6;
  Radio.data[6] = sig7;
  Radio.write();

  delay (500);       // warten um Rundlauf der Signale / Lifelock zu vermeiden
  Radio.rxMode(7);
 }
```

eleganter wäre ein Vergleich ob die Empfangene Werte den bereits vorhanden und damit selbst gesendeten entsprechen und nur zu reagieren, wenn sie unterschiedlich sind

Progammierung der Feedback/Soundeinheiten

mit bereinigten Kommentaren

Bibliotheken aufrufen
Variablen definieren

```
#include <Nrf2401.h>          // Transceiver/Funk Bibliothek
      Nrf2401 Radio;

// Radio
      int sig00 = 0;          // Art der Nachricht
      int sig01 = 0;
      int sig02 = 0;
      int sig03 = 0;
      int sig04 = 0;
      int sig05 = 0;
      int sig06 = 0;
      int anzahl = 0;

void setup (){
 // Transceiver / Funk initialisieren
      Radio.power = 3;        // Sendeleitung am Anfang auf maximum = 3
      Radio.localAddress = 1; // eigene Adresse -  alle identische Adressen
      Radio.remoteAddress = 1;
      Radio.rxMode(7);        // 7 Byte empfangen
}
```

Voreinstellungen

Alle Einheiten im gesamten System haben die gleiche Adresse „1“ - somit ist, wie auch gewünscht, keine Adressierung einzelner Einheiten möglich !!

```
void loop (){
  while(!Radio.available()) {
     }


     Radio.read();            // Daten aus Empfangspuffer auslesen
     sig00 = Radio.data[0];   // Art der Nachricht
     sig01 = Radio.data[1];   // Uebertragene Werte
     sig02 = Radio.data[2];
     sig03 = Radio.data[3];
     sig04 = Radio.data[4];
     sig05 = Radio.data[5];
     sig06 = Radio.data[6];
```

Schleife

empfangene Signale werden weitergeleitet und der Lautsprecher, der statt der Led an einem der pwm-Ausgänge angeschlossen ist, gibt dementsprechende Geräusche von sich, die als Feedback vom Nutzer interpretierbar sind

```
     // Notfallsignale mehrere Male weiterleiten, andere nur einmal
     if ((sig00 == 99) || (sig00 == 88)) {
      anzahl = 10;
     analogWrite(11, 200);
     delay (500);
     analogWrite(11,0);
     }

     if (sig00 == 77) {
```

```
anzahl = 1;
analogWrite(11, 50);
delay (20);
analogWrite(11,0);
}

if (sig00 == 33) {
anzahl = 1;
analogWrite(11, 150);
delay (20);
analogWrite(11,0) ;
}

if (sig00 == 21) {
anzahl = 1;
analogWrite(11, 30);
delay (200);
analogWrite(11,0)  ;
}

if (sig00 == 31) {
anzahl = 1;
analogWrite(11, 20);
delay (200);
analogWrite(11,0);
}

if (sig00 == 11) {
anzahl = 1;
analogWrite(11, 180);
delay (10);
analogWrite(11,0);
delay (10);
analogWrite(11, 180);
delay (10);
analogWrite(11,0) ;
}

if (sig00 == 12) {
anzahl = 1;
analogWrite(11, 150);
delay (10);
analogWrite(11,0);
delay (10);
analogWrite(11, 110);
delay (10);
analogWrite(11,0) ;
}
```

```
// Signale weiterleiten - Signale von Mobilmodul nicht weiterleiten, diese sollen
nur lokal wirken

// Signale von Mobilmodul nicht weiterleiten, diese sollen nur lokal wirken
     if ((sig00 != 31) && (sig00 != 21) && (sig00 != 20)){

// Funk weiterleiten, die letzte Zahl gibt die Anzahl der Wiederholungen
      funk (sig00,sig01,sig02,sig03,sig04,sig05,sig06, anzahl);
     }

} // Ende loop


// FUNK ++++++++++++++++++++++++++++++++++++
       void funk (int sigy00, int sigy01,int sigy02,int sigy03,int sigy04,int
sigy05,int sigy06,int wiederholung) {

// wenn der erste empfangene Wert/Nachrichtentyp sigy00 gleich dem letzten
gesendeten, dann kein Funk da wahrscheinlich Rundlauf

        for (int w=0; w <= wiederholung; w++) {
         Radio.txMode(7);
           Radio.data[0] = sigy00;
           Radio.data[1] = sigy01;
           Radio.data[2] = sigy02;
           Radio.data[3] = sigy03;
           Radio.data[4] = sigy04;
           Radio.data[5] = sigy05;
           Radio.data[6] = sigy06;
           Radio.write();
          }                              // for

       Radio.rxMode(7);                  // gehe wieder auf Empfang, 7 Byte

      }                                  // nichts zurückzugeben
```

ADRRM-Progammierung der Windsensoreinheiten

mit bereinigten Kommentaren

Bibliotheken aufrufen
Variablen definieren

```
/*  Wetterfühler aussen - nur Helligkeit zum debuggen
unsigned long zeit läuft irgendwann über - Ändern!!!
 */

#include „Nrf2401.h“
Nrf2401 Radio;
int forMax = 0;                    // blink Anzahl
int PIR;
int sig1 = 0;
int sig2 = 0;
int sig3 = 0;
int sig4 = 0;
int sig5 = 0;
int sig6 = 0;
int sig7 = 0;

int radioAlt = 0;                  // letzter gesendeter Wert

unsigned long millis();
unsigned long sensorZeit = millis() + 1000;
```

Voreinstellungen

Alle Einheiten im gesamten System haben die gleiche Adresse „1“ - somit ist, wie auch gewünscht, keine Adressierung einzelner Einheiten möglich !!

```
void setup(void)
{
 Radio.localAddress = 1;           // eigene Adresse
 Radio.remoteAddress = 1;          // von Sender
 Radio.txMode(7);                  // 7 byte senden

 pinMode (8, INPUT);               // Eingabepin-Sensor
 digitalWrite (8, HIGH);           // internen Pullup schalten > Pin HIGH
}
```

Schleife

lese den Windsensor aus und sende Gefahrensignal, wenn zuviel Wind

```
void loop(void)
{
  int sensorHell = analogRead (0);
  int sensorTemp = analogRead (3);
  int wind = digitalRead (8);

  if (sensorZeit < millis()) {
  // Check Wind
    if (wind == LOW) {
    analogWrite (9, 200);
    int sig1 = 88;
    int sig2 = 88;
    int sig3 = 88;
    int sig4 = 88;
```

```
    int sig5 = 88;
    int sig6 = 88;
    int sig7 = 88;

// dann Sende 88 als Windgefahr
    Radio.txMode(7);
    Radio.data[0] = sig1;
    Radio.data[1] = sig2;
    Radio.data[2] = sig3;
    Radio.data[3] = sig4;
    Radio.data[4] = sig5;
    Radio.data[5] = sig6;
    Radio.data[6] = sig7;
    Radio.write();

        analogWrite(11, 200);       // blinke in einer Farbe //
    }


// Check Hell
  if (sensorHell <=800) {           // wenn der PIR Detektion

  Radio.txMode(7);                  // auf senden gehen und 55 senden //
  Radio.data [0] = 77;              // 77 Messwerte im Toleranzbereich
  Radio.data [1] = 77;
  Radio.data [2] = 77;
  Radio.data [3] = 77;
  Radio.data [4] = 77;
  Radio.data [5] = 77;
  Radio.data [6] = 77;
  Radio.write();                    // raus in den Aether mit dem Kram
  // fertig mit senden

  analogWrite(10, 200);             // blinke in einer Farbe
  delay (100);                      // kurz anlassen, dann abschalten

  analogWrite(10, 0);

  sensorZeit = millis() + 1000;

  }                                 // ende if
  }                                 // sensorZeit

  analogWrite(10, 0);
  analogWrite(9, 50);
  analogWrite(11, 0);
}                                   // LOOP
```

ADRRM-Progammierung der evolutionären Spendereinheit - getrennter Evolutionsversuch

mit bereinigten Kommentaren

Bibliotheken aufrufen
Variablen definieren

```
/*  ADRRM-Spender
Sendet in kurzen Abständen das Appellsignal und danach evolutionäre Schwel-
lenwerte
 */

#include „Nrf2401.h“
Nrf2401 Radio;
```

Voreinstellungen

Alle Einheiten im gesamten System haben die gleiche Adresse „1“ - somit ist, wie auch gewünscht, keine Adressierung einzelner Einheiten möglich !!

```
void setup(void)

{
 Radio.localAddress = 1; // eigene Adresse //
 Radio.remoteAddress = 1; // von Sender //
 Radio.txMode(7); // 1 byte senden //
 Radio.power = 3; // Sendeleistung begrenzen
}
```

Schleife

Sende Appellsignal 55 zum Start

```
void loop(void)

{

 // erst Appell starten


 Radio.txMode(7);

 Radio.data[0] = 55;
 Radio.data[1] = 0;
 Radio.data[2] = 0;
 Radio.data[3] = 0;
 Radio.data[4] = 0;
 Radio.data[5] = 0;
 Radio.data[6] = 0;
 Radio.write();


 delay (300);
```

sende Evolutionskennung 66

kürze Werte auf einen Bereich < 254

mutiere Werte durch Addition eines Zufallswertes

negativen Werte durch Multiplikation mit -1 in positiven Wert umrechnen und durch 100 dividieren;
auch Emprfängerseite wieder mit 100 multipliziert

mit Led blinken, um Sendung anzuzeigen

5 Sekunden warten

```
Radio.data[0] = 66;
//Werte zum Senden durch 10 dividieren,
auf Empfängerseite dann wieder mit 10 multiplizieren
Radio.data[1] = (500 + random (-20, 21)) / 10;
Radio.data[2] = (900 + random (-20, 21)) / 10;
Radio.data[3] = 130 + random (-5, 6);
Radio.data[4] = 65 + random (-5, 6);
Radio.data[5] = ((-5000 + random (-20, 20)) / 100) * -1 // negt. in positiv
Radio.data[6] = (5000 + random (-20, 20)) / 100;
Radio.write();


   analogWrite(9, 200); // blinke in einer Farbe /
   delay (100); // kurz anlassen, dann abschalten //
   analogWrite(9, 0);


delay (5000);

}
```

9 Verzeichnisse

9.1 Literaturverzeichnis

ABUBAKR, Ibrahim: Ökosysteme, in: Komplexe Adaptive Systeme. Beiträge des Instituts für Umweltsystemforschung der Universität Osnabrück, MATTHIES, Michael; PAHL-WOSTL, Claudia; EBENHÖH, Eva (Hrsg.), 27, Osnabrück 2003.

BRÄUER, Holm: Fuzzy Logic und Wahrscheinlichkeit. Dresden 2006.

FALLIERE, Nicolas; O MURCHU, Liam; CHIEN, Eric: W32.Stuxnet Dossier. Version 1.3, Cupertino 2010.

FRINGS, Stephan; MULLER, Werner A.: Tier- Und Humanphysiologie: Eine Einfuhrung. Heidelberg 2009.

GAGE, Nathaniel Lees; BERLINER, David C.: Pädagogische Psychologie. 5. Auflage, München 1996.

GANDOLFI, Alberto: Menschen und Ameisen. Zürich 2001.

GIANNETOS, Thanassis; DIMITRIOOU, Tassos; PRASAD, Neeli R.: Weaponizing Wireless Networks: An Attack Tool for Launching Attacks against Sensor Networks. Vortrag auf der Black-Hat europe Konferenz Barcelona 2010, Barcelona 2010.

HASE, Christopher: Schwarmbasiertes Multipath-Routing in Sensornetzen. Norderstedt 2006.

HERMEY, Guido et al.: Der Experimentator: Neurowissenschaften. Heidelberg 2010.

HÖLLDOBLER, Bert; WILSON,Edward: The Ants. Harvard 1990

HOLLAND, John Henry: Studying Complex Adaptive Systems, in: Journal of Systems Science and Complexity, 19, Berlin 2006.

JOHNSON, Steven: Emergence. New York 2004.

KIRCHNER, Walter: Die Ameisen: Biolgie und Verhalten. München 2001.

KLINKE, Rainer; BAUMANN Rosemarie: Physiologie. Stuttgart 1994.

KOCH, Peter: Weniger Kohlendioxidausstoß durch Rechenzentren mit energiesparenden Kühlsystemen. 2009.

KULYK, Roman: Smart Grid. Taking our cue from nature, 2009.

LÄMMER, Stefan: Reglerentwurf zur dezentralen Online-Steuerung von Lichtsignalanlagen in Straßennetzwerken. Dissertation, Fakultät Verkehrswissenschaften "Friedrich List", Technischen Universität Dresden, Dresden 2007.

LOHMANN, Dieter: Klimaanlagen für Termiten-Türme. in: scinexx, Düsseldorf 2006.

LORENZ, Edward N.: Deterministic Nonperidoc Flow, in Journal of the Atmospheric Sciences, 20, Boston 1963.

MARX, Christy: Grace Hopper. The First Woman to Program the First Computer in the United States. New York 2004,

METZLER, R. et al.: Synchronisation neuronaler Netzwerke, in: Physical Review E, 62, 2000.

MEYER, Martin: Signalverarbeitung: Analoge und digitale Signale, Systeme und Filter. Wiesbaden 2009.

MIT Center for Collective Intelligence: Handbook of Collective Intelligence. Cambridge 2011.

MITTERBAUER, Josef: Behavior Recognition and Prediction in Building Automation Systems. TU-Wien, Fakultät für Informatik, Wien 2009.

MURPHY, Nancey C. et al.:Did my neurons make me do it? Oxford 2007.

NORDMANN, Alfred; SCHMIEDE, Rudi: Ambient Intelligence. Ethische und gesellschaftliche Herausforderungen, in:Thema Forschung - Ambient Web, 1, 2007.

SAUERMOST, Rolf; FREUDIG, Doris et al. [Redaktion]: Lexikon der Biologie. Band 12-Resolvase bis Simvastatin, Heidelberg 2002.

SCHMICKL, Thomas: Sammeln, Verteilen und Bewerten von Informationen. Verteilte Intelligenz in einem Bienen Volk, in: FACTS Die Informationsgesellschaft, 1, Wien 2003.

SCHREINER, Rüdiger Schreiner: Computernetzwerke: von den Grundlagen zur Funktion und Anwendung. München 2009.

SCOTT et al.: A Chemosensory Gene Family Encoding Candidate Gustatory and Olfactory Receptors in Drosophila. Cell, 104, New York 2001.

SEDLACEK, Klaus-Dieter et al.: Emergenz: Strukturen der Selbstorganisation in Natur und Technik. Norderstedt 2010.

STACHOWIAK, Herbert:Allgemeine Modelltheorie. Wien, 1973.

STARKE, Sandra: Lernen und Lernumgebung- Das Multi-Agenten-System“School Agency“. Norderstedt 2002.

STRÖM, Pär: Die Überwachungsmafia. Wien, 2005.

SUGAHARA, Michio; SAKAMOTO, Fumio: Heat and carbon dioxide generated by honeybees jointly act to kill hornets, in: Naturwissenschaften, 96, Heidelberg 2009.

SYCARA, Katia P.: Multiagent Systems, in AI Magazine 19/2. Menlo Park 1998.

TASS. Peter A.: Hilfe bei Parkinson und Bewegungsstörungen. Jülich 2006.

TOSH, Colin; RUXTON, Graeme D.: Modelling Perception with Artificial Neural Networks. Cambridge 2010.

WEGENER, Axel; HELLBRÜCK, Horst et.al: Designing a Decentralized Traffic Information System AutoNomos, in: Kommunikation in Verteilten Systemen (KiVS), Hrsg. DAVID, Klaus, GHEIS, Kurt, Berlin 2010.

9.2 Internetverzeichnis

Academic dictionaries and encyclopedias: Chaostheorie. (http://de.academic.ru/dic.nsf/dewiki/247319), 10. Juni 2011.

Arduino Community, (http://www.arduino.cc/), 23. Mai 2011.

BESTEHORN, Michael: Deterministisches Chaos. (https://www.physik.tu-cottbus.de/physik/tp2/subdir/schuler/det_chaos.pdf), 30. Mai 2011, S. 35.

Bundesministerium der Justiz: Bundesdatenschutzgesetz. (http://www.gesetze-im-internet.de/bdsg_1990/__3a.html), 26. April 2011.

Bundesministerium der Justiz: Bundesdatenschutzgesetz. (http://www.gesetze-im-internet.de/bdsg_1990/__4.html), 26. April 2011.

Bundesministerium der Justiz: Bundesdatenschutzgesetz. (http://www.gesetze-im-internet.de/bdsg_1990/__4a.html), 26. April 2011

Cyber Ark, (http://www.cyber-ark.com/news-events/pr_20080619.asp), 12. Februar 2009.

Digitalstrom, (http://www.digitalstrom.org/), 3. Juni 2011.

DOWRSCHAK, Manfred; KELLER Keller: Supermacht im Untergrund, in: Der Spiegel, 17, 2002, (http://wissen.spiegel.de/wissen/image/show.html?did=22151068&aref=image029/E0217/SCSP200201702040205.pdf&thumb=false), 20. Februar 2011.

EIB/KNX, (http://www.knx.de/), 3. Juni 2011.

Fraunhofer inHaus, (http://www.i-r.de/downloads/Fraunhofer-inHaus.pdf), 3. Juni 2011.

Kaspersky Labs GmbH, (http://www.kaspersky.com/de/news?id=207566365), 27. April 2011.

LON, (http://www.lonmark.de/), 3. Juni 2011.

pcmag.com ececlopedia: distributed intelligence. (http://www.pcmag.com/encyclopedia_term/0,2542,t=distributed+intelligence&i=41566,00.asp#), 16. Februar. 2009.

Sparkfun Electronics: How Far Does It Go? (http://www.sparkfun.com/tutorials/48), 2. Juni 2011.

Sparkfun Electronics, (http://www.sparkfun.com/), 23. Mai 2011.

Sparkfun Datenblatt: Single chip 2.4 GHz Transceiver nRF2401A. (http://www.sparkfun.com/datasheets/IC/nRF2401A.pdf), 27. Mai. 2011.

Regen Energy Corp., (http://www.regenenergy.com/default.htm), 1. Mai 2011.

RWE Smart Home, (http://www.rwe.com/web/cms/de/455660/rwe/innovationen/energieanwendung/smarthome/), 3. Juni 2011.

Siemens APOGEE, (http://www.buildingtechnologies.siemens.com/bt/us/SiteCollectionDocuments/sbt_internet_us/products-systems/building-automation/apogee/siemensapogeescalable.pdf), 3. Juni 2011.

Smartlighting, (http://www.bu.edu/smartlighting/), 3. Juni 2011.

Verein zur Förderung des öffentlichen bewegten und unbewegten Datenverkehrs e.V.: Bürgerrechtsorganisationen: Straßen-Total-überwachungs-Vertrag mit TollCollect kündigen. (www.foebud.org/misc/StoppTollCollect.pdf), 18. Mai 2011.

KIM, Martin (Autor): Klangbild. (http://vimeo.com/22550950), 3. Juni 2011.

Wikipedia: Nichtlineare_Systeme. (http://de.wikipedia.org/wiki/Nichtlineare_Systeme), 25. Februar 2011.

ZigBee, (http://www.zigbee.org/), 3. Juni 2011.

10 Abbildungsverzeichnis

11 Wissenschaftlicher Werdegang

1989	Allgemeine Hochschulreife, *Friedrich-Dessauer Gymnasium*, Frankfurt am Main
1990-1999	Studium der Architektur - *Technischen Universität Darmstadt*
1995-1996	Studium der Architektur - *Universita Degli Studi*, Florenz Studium der Bildenden Künste - *Accademia di Belle Arti*, Florenz
1999	Diplomarbeit und Diplom in Architektur - *Technischen Universität Darmstadt*
2000-2001	Wissenschaftlicher Mitarbeiter - *Technische Universität Darmstadt* (Fachgebiet Entwerfen und Gebäudetechnologie, Prof. Karl-Heinz Petzinka)
2002-2005	Wissenschaftlicher Mitarbeiter mit Lehrauftrag - *Technische Universität Darmstadt* (Fachgebiet Plastisches Gestalten, Prof. h.c./i.V. Ariel Auslender)
2005	Annahme als Doktorand FB 15 Architektur - *Technische Universität Darmstadt* (Professor Karl-Heinz Petzinka, Professor Manfred Hegger)
2007	Lehrauftrag - *Technische Universität Darmstadt* (Fachgebiet Archäologie, Prof. Franziska Lang)
2011	Abgabe der Dissertation 18. Oktober Disputation
	Lehrauftrag - *Technische Universität Darmstadt* (Fachgebiet Entwerfen und Industrielle Methoden der Hochbaukonstruktion Prof. i. V. Wolfgang Hinkfoth)

12 Erklärungen

Dipl.-Ing. Martin Kim
Mühlstraße 46
64283 Darmstadt

Hiermit versichere ich, dass ich die an der Technischen Universität Darmstadt eingereichte Dissertation

Automation nach dem ADRRM-System
Adressloses Dezentrales Reichweitenbeschränktes Redundanzbasiertes Minimalinformationssystem zur Automation von Gebäuden und Urbanen Räumen

an dieser Universität erstmals einreiche und an keiner anderen Universität oder Hochschule vorgelegt habe.

Martin Kim

Darmstadt, den

Dipl.-Ing. Martin Kim
Mühlstraße 46
64283 Darmstadt

Hiermit versichere ich, dass die an der Technischen Universität Darmstadt eingereichte Dissertation

Automation nach dem ADRRM-System
Adressloses Dezentrales Reichweitenbeschränktes Redundanzbasiertes Minimalinformationssystem zur Automation von Gebäuden und Urbanen Räumen

soweit nicht anders gekennzeichnet das Ergebnis meiner eigenständigen Arbeit ist.

Martin Kim

Darmstadt, den

13 Abstract

Abstract deutsch

Diese Arbeit beschäftigt sich mit einem System zur Automatisierung von öffentlichen und halböffentlichen Räumen mit dem Fokus auf dem Schutz der Privatsphäre der Nutzer. Herkömmliche Systeme gefährden die Privatsphäre der Nutzer, da sie zur Erfüllung ihrer Steuerungsaufgaben Nutzerdaten zentral erfassen müssen und somit alle Nutzerdaten auch zentral abrufbar sind. Ein zuverlässiger Schutz der Privatsphäre kann hingegen in vollständig dezentralen Systemen gewährleistet werden, da diese keine zentrale Datenerfassung benötigen und daher keinen Zugriff auf Nutzerdaten erlauben. Dies Arbeit beschreibt den Aufbau solcher dezentralen Systeme zur Automatisierung von öffentlichen und halböffentlichen Räumen und weißt deren grundsätzliche Funktionstüchtigkeit nach.

Abstract english

This work is concerned with a system for the automation for public and semi-pulbic spaces with a focus on the protection of the user privacy. Common systems threaten the user privacy because they have to collect user data centrally to fulfil their tasks and therefore all user data is also centrally accessable. A reliable protection of the user data can be provided in complet decentral systems as these systems do not need to collect user data centrally and therefore the user data is inaccessable in these systems. This work descibes the structure of decentral systems for the automation for public and semi-pulbic spaces and proofs their general functionality.